A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

) III, NÚMERO 36

975 Pts.



ACKAGES Y CLASES ANIMADAS

TELIGENCIA ARTIFICIAL
Earemos vida en un PC

ROGRAMACIÓN X-WINDOW
Ver SVGA para XFree86

Y ADEMÁS...

- El Pentium II
- Programación de Emuladores
- ACCESS y Visual Basic
- Oracle 8

CONTENIDO DEL CD-ROM

- JAVA: JDK 1.1.3, Java Workshop 2.0, etc
- Netscape Communicator 4.0
- Proxies para 95 y NT: Wingate 2.0, WebPop 1.0, Quickland 97, etc.
- Demonios para NT
- Módulos Active-X y VB

HTML DINÁMICO
Con Netscape Communicator y JavaScript

REDES LOCALES
Protección de datos



TOWER

SÓLO PROGRAMADORES

Número 36
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es

Directora Adjunta
Amelia Noguera
anoguera@towercom.es

Colaboradores
Enrique Díaz, Arsenio Molinero, Carlos Álvaro, Enrique de la Lastra, Francisco José Abad Cerdá, Jose Antonio Collar, Javier Sarsa, Oscar Prieto Blanco, Ernesto Schmitz, Alejandro Reyero, Emilio Postigo Rian, Chema Álvarez, J. Antonio Mendoza, Jorge Delgado Mendoza.

Jefe de Diseño
Fernando García Santamaría

Maquetación
Clara Francés

Tratamiento de Imagen
María Arce Giménez

Imagen de Portada
Fernando Núñez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
eriva@towercom.es

Publicidad
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Filmación
Megatipo

Impresión
G. Reunidas

Distribución
SGEL

Director General
Antonio M. Ferrer Abelló

Director Financiero
Francisco García Díaz de Liaño

Director de Producción
Carlos Peropadre

Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792

EDITORIAL

El saber estival

Hay quien dice que la del verano es una época poco propicia para iniciar nuevos proyectos. Que bajo el sol justiciero de lo más fuerte de la canícula, después de un julio aciago y en un agosto indolente, si ya es difícil pensar algo, tanto más lo será pensar algo nuevo. En esto hay algo de razón, pero también es cierto que en el vacío intelectual de las playas abarrotadas, en el vacío existencial que surge de entre los valles y las montañas, se dan infinitos momentos de, para qué nos vamos a engañar, soberano y ancestral aburrimiento. Y qué mejor que llenar ese perezoso aburrimiento con un conocimiento nuevo, en nuestro caso un nuevo programa o, por qué no, un nuevo lenguaje de programación.

Pero, ¿cuál? ¿Qué lenguaje desconocido nos va a ser útil después, cuando la canícula se retire hasta el año próximo y las primeras hojas decadentes del otoño anuncien una nueva temporada laboral? La respuesta es clara: JAVA. El motivo también lo es: las grandes compañías están dando un papel preponderante a este lenguaje, gracias a sus cualidades, muy por encima de otros que están empezando a caer en un lento y progresivo desuso. Así pues, JAVA es un presente y un futuro que los informáticos no debemos dejar de lado. Por eso es el tema de portada y nuestro artículo estrella. En él se examinan los *packages*, las clases anidadas y los modificadores de acceso de este lenguaje de programación, tres de las características que lo definen.

Otros artículos podrán cautivar al lector y salvarlo de los rigores del verano. Se trata, por ejemplo, de la programación de tarjetas de vídeo en Linux, de la programación de emuladores o de la inteligencia artificial. Porque, cómo no, seguimos a vueltas con la inteligencia artificial. De hecho, los aficionados a este tema encontrarán en el CD-ROM un interesante programa que imita las características de una sociedad, aunque sólo sea de plantas y hormigas.

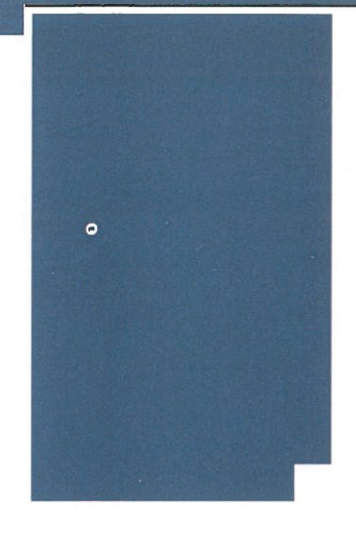
En este verano que ha resultado un poco más oscuro y más triste que los demás, pero también quizá más esperanzador, confiamos, como siempre, en que, con la ayuda de nuestras páginas, podáis aprender mucho y bueno. Y reencontrarnos después del verano renovados. Os adelantamos que el mes de septiembre va a traernos sorpresas muy interesantes: ¿qué os parecería un especial de Linux?; esperamos que no os lo perdáis.

Porque en la redacción seguiremos trabajando duro para vosotros.

Tenéis a vuestra disposición la dirección de correo electrónico de la revista: **solop@towercom.es,** donde podéis enviar vuestras sugerencias sobre los contenidos, los temas que queráis que incluyamos y los productos que os gustaría que distribuyéramos con el CD-ROM, que todos los meses regalamos.
Sois nuestros mejores colaboradores.



SÓLO PROGRAMADORES



Noticias

OPENINGRES 2.0
Nueva versión del sistema de gestión de base de datos de Computer Associates

Este sistema de gestión de bases de datos lleva mucho tiempo funcionando dentro de las empresas que necesitan mantener su información actualizada, doce años nada más y nada menos son los que han transcurrido desde que apareció la primera versión y es ahora cuando desde Computer Associates se ha decidido actualizar la herramienta para abarcar la tecnología del momento.

Junto con nuevas mejoras incluidas por solicitud de los usuarios, se añaden funcionalidades de gestión y mejoras en el rendimiento, mediante las cuales se consigue incrementar de forma significativa la capacidad de conectividad y replicación, simplificando la creación y gestión de las aplicaciones basadas en la Web.

OpenIngress es un sistema multiplataforma de gestión de bases de datos para procesos empresariales de misión crítica, y como mejoras incluye las siguientes:

- bloqueo a nivel de fila especificado a nivel de tabla o de usuario
- soporte para Java
- mayor conectividad en la web o través del soporte de APIs nativos para los servidores de Microsoft, Netscape y Spyglass
- mejoras en la tecnología de replicación

La mayor parte de las localizaciones son controladas por bases de datos, de tal forma que si habitualmente una página requería un programa que integrara consultas SQL y cuando se producía un cambio en una página Web, el programa debía cambiar y probar dicha página para ponerla on-line, con la versión 2.0 de Open Ingress, ya no es necesario manejar estos programas intermedios.

Además, el producto incluye un macroprocesador que permite a los programadores integrar sentencias SQL en una página HTML, permitiendo al servidor Web procesar estas consultas en tiempo de ejecución, simplificando la gestión de los contenidos a los usuarios.

La velocidad de replicación ha sido incrementada, al trasladar el núcleo del sistema de captura de la replicación al motor del servidor de la base de datos.

Otras de las nuevas características del producto incluyen una mejor conectividad entre plataformas cruzadas y mejor soporte de red, almacenamiento de tamaños de página más flexible, nuevas reglas en el ámbito de sentencia (*triggers*), así como mejoras sustanciales en el optimizador estadístico para realizar una gestión de consultas mucho más eficaz. En cuanto a la gestión, se ha intentado mejorar la gestión del espacio en disco, se han diseñado nuevos algoritmos para la gestión de caché y también se soportan datos espaciales.

SOLARIS MILLENIUM
Nueva versión del conocido Solaris para la informática de red

Sun Microsystems, Inc. anuncia que ya están disponibles en el mercado las mejoras para su entorno operativo Solaris. La web sigue siendo la protagonista completa de dichas mejoras, englobando áreas como los grupos de trabajo en internet, intranets, la informática empresarial y los equipos de sobremesa de gran potencia.

La nueva versión se dirige especialmente a los profesionales de la tecnología de la información, directores de líneas de negocio y proveedores de servicios Internet (los llamados ISP), y a los ya mencionados usuarios de equipos de sobremesa de gran potencia. Solaris se toma como base para el *Webtone* (comparable al tono telefónico actual), ya que es posible el acceso al mismo y el uso de dispositivos conectados a la red de todo tipo y tamaño. Solaris funciona tanto en ordenadores SPARC como en los basados en Intel.

La versión de Solaris con mejoras para la Web permite trabajar en cualquier red y con cualquier dispositivo conectado a dicha red, incluyendo Mac, Windows, NCs o Unix, ya que se cumplen todos los estándares HTTP, IIOP y WebNFS. Además, Java IDL (*Interface Definition Layer*) permite a los clientes Java acceder a las aplicaciones ya existentes.

Como nota práctica de la noticia podemos decir que desde Sun señalan a Solaris como el entorno operativo más rápido del mundo para aplicaciones Java. En esta versión de Solaris se incorpora Java como parte integral del entorno operativo. Junto con el navegador HotJava, la nueva versión de Solaris incluye las siguientes herramientas:

- Java Virtual Machine
- Jit Compiler
- Integrated Java APIs

SE FIRMA EL ACUERDO SOBRE EL ESTÁNDAR HTML 4.0
Netscape y Microsoft adoptan un mismo estándar

De forma definitiva se ha firmado un acuerdo entre los dos gigantes de la Web: Microsoft y Netscape para adoptar un estándar único en el HTML 4.0.

Este acuerdo intentará acabar con la que casi se ha llegado a convertir en una guerra de navegadores, al servir como base para una unificación de los criterios en el desarrollo de páginas Web. Las características de este estándar son las siguientes:

- mejores formularios, con las opciones más conocidas de Windows y Mac.
- mejores tablas, con scroll, encabezamientos fijos, etc.
- mejores frames.
- facilidad de programación, proporcionando una forma estándar para realizar la inserción de objetos y scripts.

Breves

KEYFLOW 2.0 PARA MICROSOFT EXCHANGE SERVER

Suricata, S.A. anuncia para después del verano la aparición de Keyflow 2.0. El producto es un programa de flujos de trabajo para la automatización de procesos empresariales que está diseñado para la plataforma Microsoft Exchange Server.

El programa permite mejorar los ciclos de producto, incrementando la productividad, al utilizar la infraestructura de la red, las agendas comunes, la gestión de objetos y la seguridad de Microsoft Exchange Server, aportando herramientas de diseño gráfico, interfaces de usuario e instrumentos para la presentación de los objetos.

OPENCONNECT SYSTEMS Y ORACLE OFRECEN ACCESO INMEDIATO A SISTEMAS HEREDADOS A TRAVÉS DE INTERNET

Ambas compañías han firmado un acuerdo para combinar sus tecnologías de forma que se permita el acceso instantáneo a sistemas heredados por todo el mundo mediante la red Internet. Gracias a este acuerdo, cualquier explorador estándar podrá acceder a cualquier aplicación estándar y extenderla hacia la Web, sin riesgos ni modificaciones en el sistema heredado y sin necesidad de programar.

Gracias a este acuerdo ambas compañías unen sus tecnologías para cubrir las necesidades del sector de mercado del comercio electrónico. Para asistir a demostraciones de productos OpenConnect Systems a través de Internet, solicite la clave de acceso en el número: 07-31 30 241 70 70.

Nueva versión de 4D para Windows y MacOs
ACI PRESENTA LA VERSIÓN 6

4D es un gestor de bases de datos relacionales independiente de la plataforma, que utiliza la tecnología orientada a objetos permitiendo el desarrollo de aplicaciones de bases de datos modulares y permite un control total del funcionamiento de la aplicación en plataformas diferentes.

La nueva versión del producto dispone de una nueva interfaz de usuario adaptada a los requisitos de las interfaces de Windows 95, NT 4.0 y MacOs 8.

Las mejoras son varias e incluyen una serie de paletas de propiedades que tiene cada objeto de la estructura (tablas, campos, relaciones, variables, gráficos y texto), lo cual facilita el acceso a los objetos; todos los objetos aparecen en una ventana y se encuentran organizados por temas; existen asistentes para la creación de formatos de pantalla para ayudar a los usuarios en la fase de diseño; se han añadido objetos nuevos de interfaz que se adaptan a los distintos sistemas operativos para los que está diseñado 4D; y se permite arrastrar y soltar los objetos en los modos de Estructura y Usuario. Además, el debugger ha sufrido una serie de modificaciones para mejorarlo sustancialmente.

Este producto incluye un lenguaje de programación muy potente que engloba más de 500 comandos y constantes para facilitar la sustitución de líneas de código que componen los programas para realizar las operaciones sobre la base de datos.

4D se basa en su propio sistema multitarea, independiente del sistema operativo, lo cual permite al usuario realizar diversas tareas a la vez. Para más información sobre el producto, consultar al distribuidor al núermo de teléfono (93) 4193358, o bien a través de e-mail: ambit@ambit.seker.es.

SILICOM GRAPHICS SOPORTA ORACLE8.
La versión del sistema de gestión de base de datos de Oracle

La empresa presenta servidores empresariales especializados y de gran tamaño creados sobre la última versión de Oracle. De esta forma Silicon Graphics demuestra su apoyo a Oracle8, la base de datos de nueva generación de Oracle para entornos distribuidos.

Oracle8 es una importante herramienta para gestionar sobre redes informáticas grandes cantidades de información, en aplicaciones Data Warehouse con tecnología orientada a objetos. Además, sus características de paralelismo automático y escalabilidad la convierten en un producto estrella para SG.

NONSTOP SQL/MX

Tandem ha presentado su sistema gestor de base de datos paralelo y abierto, NonStop SQL/MX, formando parte de su oferta de NonStop Software para informática de cluster de misión crítica.

NonStop SQL/MX funciona sobre cualquier sistema basado en Windows NT Server e Intel (multiprocesador simétrico o configuración en cluster), y sobre los servidores NonStop Himalaya de Tandem. La nueva base de datos es independiente de la plataforma y amplía las prestaciones y disponibilidad de las bases de datos de Tandem a la plataforma Windows NT Server al tiempo que añade data mining avanzado, gestión de objetos y optimización de consultas.

En la actualidad han sido realizadas varias demostraciones de esta nueva base de datos mediante la utilización de una base de datos de 2 terabyteas y una tabla de 30.000 millones de filas operando sobre un cluster Windows NT de 64 procesadores, obteniendo de esta forma un paralelismo masivo. NonStop SQL/MX es capaz de mantener una única imagen de la base de datos a través de todos los servidores en un cluster, permitiendo de esta forma que la aplicación sea gestionada del mismo modo que si ésta residiera en un único servidor.

El producto soporta la tecnología DAtaBlade de Informix para permitir que tipos de datos complejos como voz, vídeo, y otros objetos multimedia sean incluidos en un amplio proceso de consultas y en un entorno de Data Mining.

Progress Sotware adopta el lenguaje Java para el desarrollo de aplicaciones empresariales

Para ello ha adquirido recientemente la compañía Apptivity Corporation que fue pionera en comercializar las herramientas de bases de datos en Java, permitiendo a los desarrolladores construir e implementar aplicaciones escalables intranet/extranet de alto rendimiento.

Java está adquiriendo día a día más importancia dentro del desarrollo y la implementación de aplicaciones empresariales y desde Progress Software se unen a esta tendencia.

La compañía adquirida ha desarrollado una herramienta muy optimizada para la construcción de aplicaciones con clientes Java y servidores Java, de forma altamente innovadora. Apttivity es la primera compañia que aprovechó al máximo las características de este lenguaje, desarrollando un entorno de desarrollo denominado Visual Apptivity que permite la construcción de aplicaciones *thin clien* interactivas de alto rendimiento (menores de 250 kb) optimizadas para las intranets y extranets corporativas.

La arquitectura multinivel del producto permite el desarrollo de la lógica del servidor que se puede implementar en el servidor de aplicaciones Apptivity mejorando de forma importante el rendimiento, simplificando el mantenimiento y reduciendo los costes del ciclo de vida de las aplicaciones. El servidor es multitarea y soporta la funcionalidad *load balancing* para escalar a un gran número de usuarios concurrentemente. El producto permite construir aplicaciones que se pueden conectar a distintas bases de datos.

Para más información consultar la dirección: http://www.progress.com.

BBV, ALCAMPO E INFOMALL DE TSAI (TELEFÓNICA DE SISTEMAS) CON NETSCAPE

Para operar con sus sistemas más avanzados

BBV y TSAI ha desarrollado para el área de banca un servicio denominado BBV net, utilizando productos y servicios de Netscape. El BBV net permite al cliente del banco acceder mediante Infovía de Telefónica o Internet a los servicios bancarios desde el ordenador de su propio domicilio.

Gracias a este servicio se puede disponer initerrumpidamente de todo tipo de información y es posible realizar traspasos de cuentas dentro del mismo banco, trasferencias a otros bancos y cambios de claves para entrar en BBV net. El objetivo es claro, ofrecer un servicio personalizado al cliente y una atención más eficaz y crear una plataforma tecnológica que permita al banco evolucionar conforme lo haga la tecnología internet.

Para obtener más información sobre este servicio puede llamar al 055, si se es usuario de Infovía, o bien mediante la dirección http://www.bbv.es/bbvnet, si se accede a través de Internet.

Breves

PREMIO NUEVAS TECNOLOGÍAS EN EL TRABAJO

Bit, empresa especializada en formación informática, convoca el premio para celebrar su treinta aniversario. Todos aquellos interesados en participar en dicho Premio deberán presentar un trabajo escrito sobre la implantación de las Nuevas Tecnologías en el Trabajo. Los artículos no deben haber sido publicados con anterioridad y se valorará la originalidad de la exposición en los artículos que podrán tratar sobre temas reales o imaginarios.

Los premios consistirán en becas de formación en los centros de Bit en Barcelona, o a través de Internet, un diploma acreditativo y una lote de libros técnicos a elegir del fondo editorial de Marcombo.

Para más información: (93) 209 29 66 o en la Web http://www.bit.es

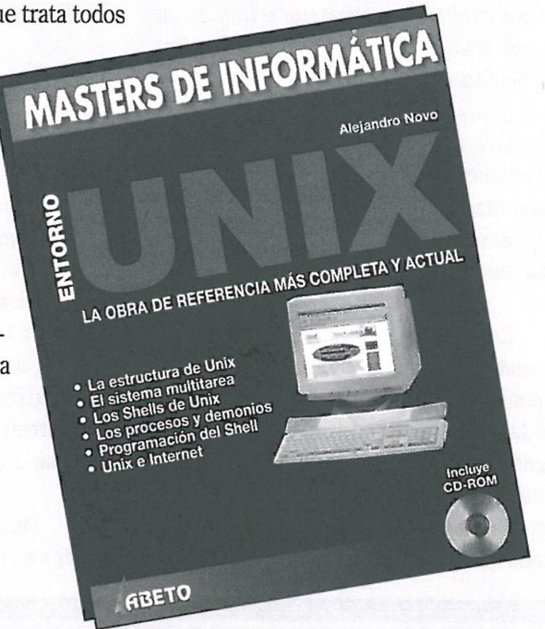
DIGITAL CON LOS CIBER RESTAURANTES

Situado en la calle Fuencarral, el restaurante ofrece comida tradicional casera junto con los más variados servicios informáticos. Carmen, la cadena de restaurantes con participación de RENFE, ONCE, Freixenet y el grupo "La taberna del Alabardero", se ocupa de la restauración, mientras que Digital se ocupa de la parte más informática.

El área informática incluye 20 nodos de acceso a Internet, con PC Digital Venturos FX de última generación, conectados en red a un servidor Prioris HX 6000 con salida a Internet, vía RDSI. También se ofrecen servicios de impresión y Escaneado, plottering de planos y una zona infantil de juegos con consolas y software Home en CD-ROM.

EL ENTORNO UNIX MASTER DE INFORMÁTICA

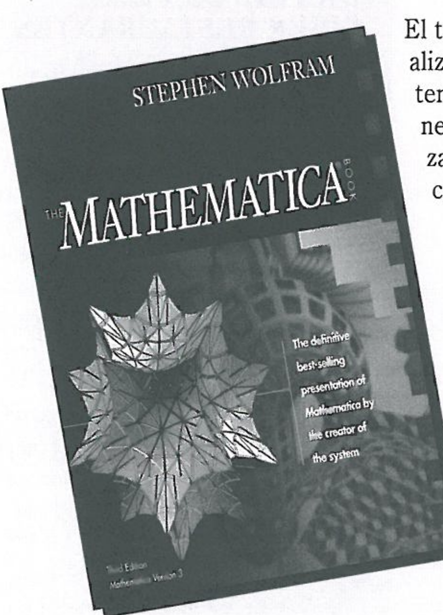
Dentro de la serie de libros "Másters de informática" se encuadra esta obra que trata todos los aspectos sobre Unix y su entorno en varias de sus variantes (SCO; Berkeley, etc.). Entre otros se profundiza en los siguientes temas: la estructura de Unix, el sistema multitarea, los procesos y los demonios, los shell de Unix, la programación de la shell, Internet y Unix, las X-Window.



Encontrará en esta obra un libro riguroso, conciso y escrito en un lenguaje sencillo, para adentrarse en el aprendizaje de Unix de forma rápida y efectiva. El autor, Alejandro Novo, consigue con *El entorno Unix* un material de referencia muy útil para avanzar en el dominio de Unix. En el CD-ROM que acompaña al libro se incluyen todos los ejemplos que aparecen en el texto, junto con programas del autor, utilidades para Unix y una versión completa de Linux.

Editorial: Abeto
Autor: Alejandro Novo
Idioma: español
Incluye CD-ROM
Precio: 4.995 ptas.

THE MATHEMATICA BOOK



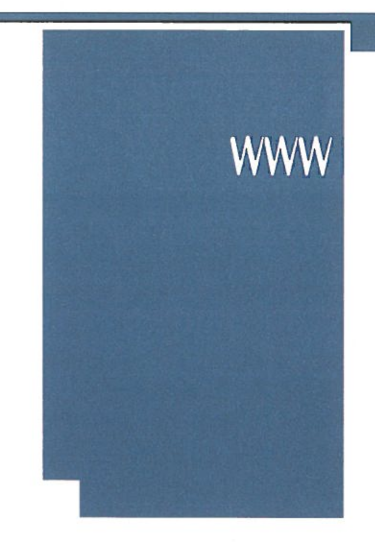
El tema principal sobre el que trata la obra, el programa Mathematica, es capaz de realizar las funciones de una calculadora, como son todo tipo de cálculos con una gran potencia, ya que incorpora los conocimientos más actuales del mundo matemático y tiene como fundamento los algoritmos de computación más precisos y rápidos, visualizando además todo tipo de gráficos basados en funciones matemáticas de una gran complejidad.

Numerosos científicos, ingenieros, médicos e investigadores utilizan este programa para realizar todo tipo de operaciones matemáticas, así como presentaciones de documentos.

A través de las 1400 páginas que forman la obra, el amplio equipo de desarrolladores de Mathematica profundiza en el conocimiento de este completísimo programa con gran precisión y claridad. Sólo un inconveniente, está escrito en inglés.

Editorial: Cambridge University Press
Idioma: inglés
1401 páginas

HTML Dinámico con Netscape Communicator y JavaScript (I)



Alejandro M. Reyero Abad
axl@las.es

El boom del HTML llegó con las versiones 3.0 de Navigator y Explorer. Todo el mundo tiene hoy día su propia página WEB, es fácil de programar, rápido y bastante vistoso, pero si bajamos a las profundidades del HTML veremos que está bastante limitado para usos profesionales, es excesivamente simple. Netscape Communicator 4.0 intenta solucionar estos problemas introduciendo un nuevo concepto, el HTML Dinámico.

Hojas de estilo

Las hojas de estilo suponen un importante avance a la hora de presentar nuestras páginas WEB, proporcionándonos mayor control sobre los elementos que las componen. Utilizando hojas de estilo podremos especificar colores a textos, incluir márgenes, alineamientos, tamaños, etc. Podremos también especificar estilos para grupos de elementos HTML. Por ejemplo, podríamos hacer que todos los párrafos de una página WEB fuesen de color amarillo. Incluso podemos declarar clases de estilos y asignar cualquier elemento que queramos a nuestra clase. Por ejemplo, podríamos crear una clase llamada VERDESUBRAYA cuyo estilo sea verde, de texto grande y con subrayado. Cualquier elemento, ya sea una cabecera, párrafo o texto en un link, podrá ser miembro de esta clase, con lo que aparecerá en la página de color verde, con texto grande y subrayado. También, por supuesto, podremos asignar estilos a elementos individuales.

Hojas de estilo en Netscape Comunicator

Netscape Communicator soporta dos tipos de hojas de estilo, las hojas de estilo en cascada (*Cascading Style Sheets* o CCS) y las hojas de estilo JavaScript.

En ambos tipos de hoja de estilo, cada atributo está asociado con una propiedad. Por ejemplo, la propiedad para fijar el margen derecho sería marginRight en JavaScript y margin-right en CCS.

Netscape Communicator soporta dos tipos de hojas de estilo: las hojas de estilo en cascada y las hojas de estilo JavaScript

La nueva y revolucionaria versión de Netscape Navigator, ahora Netscape Communicator, supone un salto cuantitativo y cualitativo, tanto en el nivel de usuario como de programador HTML/JavaScript. El nuevo concepto de HTML dinámico nos introduce en un mundo nuevo de hojas de estilo, layers y fuentes dinámicas, hasta hoy desconocido.

Esta sintaxis es la utilizada en la mayoría de los casos, esto es, los nombres de las propiedades en la sintaxis JavaScript combinan dos palabras, juntas, empezando la segunda de ellas con mayúscula, por ejemplo marginRight o textTransform. La sintaxis de las hojas de estilo en cascada utiliza un guión para separar las dos palabras de la propiedad, estando ambas en minúsculas completamente, por ejemplo margin-right o text-transform. Los nombres de las propiedades que son una sola palabra son generalmente idénticos en JavaScript o CCS, por ejemplo, color.

Para ambos tipos de hoja de estilo, cada propiedad se ve reflejada en una propiedad JavaScript, con lo que los autores de páginas WEB estarán capacitados para utilizar JavaScript a la hora de manipular las hojas de estilo.

Hojas de Estilo en Cascada

Netscape Communicator soporta el estándar para hojas de cascada que se encuentra definido en el documento de la W3.org, cuya dirección es la siguiente: http://www.W3.org/pub/WWW/TR/REC-CSS1.

Por ejemplo, la siguiente hoja de estilo en cascada hará que el tamaño de todos los párrafos (<P>) sea 20 y que su margen derecho sea de 100, además, el color de todos los <H2> será verde:

```
<STYLE TYPE="text/css">
P {font-size: 20pt; marginRigh:100pt;}
H2 {color: green;}
</STYLE>
```

Como vemos, los nombres de las propiedades y los valores para cada elemento HTML van incluidos entre paréntesis a continuación del nombre del elemento HTML. Cada conjunto propiedad/valor tendrá dos puntos separándolos y terminarán en punto y coma.

Hojas de Estilo JavaScript

Las hojas de estilo en JavaScript son una manera alternativa de definir hojas de estilo (en mi opinión más sencilla y natural que las CCS).

El mismo ejemplo que utilizamos para ver la sintaxis de las hojas de estilo CCS quedaría de la siguiente manera en JavaScript:

```
<STYLE TYPE="text/javascript">
    tags.P.fontSize=20;
    tags.P.marginRight=100;
    tags.H2.color="green";
</STYLE>
```

A partir de ahora trabajaremos con hojas de estilo en JavaScript. La conversión entre los dos tipos de hoja de estilo es trivial de todas maneras.

Herencia de estilos

Las hojas de estilo tienen elementos padre e hijo, un elemento contenido dentro de otro será su hijo. Por ejemplo, en el siguiente código HTML el elemento <HEAD> es padre del elemento <H2>, que a su vez es padre del elemento <STRONG>:

```
<HEAD>
<H2>    Todo el mundo
    <STRONG>lee</STRONG>
    Solo Programadores</H2>
</HEAD>
```

Un elemento hijo hereda las propiedades de su padre. Supongamos que el ejemplo <H2> anterior tuviera un estilo asignado de la siguiente manera:

```
<H2 CLASS="VERDESUBRAYA">
Todo el mundo <STRONG>lee</STRONG>
Solo Programadores
</H2>
```

En este caso, el elemento hijo (el <STRONG>) utilizará el estilo de su padre. Si anteriormente hubiéramos creado un estilo que especificara que el elemento <STRONG> debería ser presentado de color azul, la palabra "lee" sería presentada en color azul, ya que las propiedades de un hijo (las que están definidas) se utilizan en vez de las heredadas del padre.

La herencia comienza en el elemento más "arriba". En HTML, esto se corresponde con el elemento <HTML>, seguido de <BODY>.

Podemos también definir propiedades de estilo por defecto, definiendo estilos para el elemento <BODY>. Por ejemplo, el siguiente código hace que el texto sea de color verde:

```
<STYLE TYPE="text/javascript">
    tags.BODY.color="green";
</STYLE>
```

Si queremos que alguna porción del texto en nuestra página aparezca de color diferente al verde tendremos que definir una clase o estilo que utilice un color diferente o definir el color de la fuente directamente (<FONT COLOR="....">).

Creación de hojas de estilo

Hay varias maneras de especificar estilos utilizando hojas de estilo. Podremos crear hojas de estilo externas y linkarlas en nuestro documento o crearlas en el documento directamente. Podemos definir hojas de estilo de alguna de las siguientes maneras:

- Definiendo estilos con el TAG <STYLE> en la cabecera.
- Especificando estilos para elementos individuales.
- Definiendo clases de estilos.

- Definiendo estilos únicos con el atributo ID.
- Utilizando el método contextual().
- Definiendo hojas de estilo en ficheros externos.
- Combinando hojas de estilo.
- Definiendo estilos con el TAG <STYLE> en la cabecera

Podemos usar el TAG <STYLE> en la cabecera de un documento para definir estilos de determinados elementos en nuestra página WEB. Los estilos que definamos en la cabecera se aplicarán en todo el documento.

Por ejemplo:

```
<HTML>
<HEAD>
<TITLE>A Grand Title</TITLE>
<STYLE TYPE="text/javascript">
	tags.H2.color = "red"
</STYLE>
</HEAD>
<BODY>
	<H2>Esto es ROJO</H2>
</BODY>
```

En este ejemplo, el estilo que hemos definido en la cabecera hace que todos los TAGs <H2> que aparecen en el documento presenten su texto en color rojo.

• Especificando estilos para elementos individuales

El TAG <STYLE> puede ser utilizado para definir elementos individuales. Por ejemplo, podríamos hacer que un párrafo (y sólo ese párrafo) fuera de color rojo:

```
<BODY>
<P STYLE="color = 'red'">
	Este párrafo es ROJO</P>
<P>Este párrafo NO es ROJO</P>
</BODY>
```

• Definiendo clases de estilos

Se pueden definir clases de estilos utilizando la propiedad classes dentro del TAG <STYLE>. Por ejemplo, podríamos definir una clase llamada ROJOBOLD, a la que invocaríamos cada vez que quisiéramos que alguno de nuestros elementos fuera de color rojo y la fuente del tipo bold:

```
<HEAD>
<TITLE>Title</TITLE>
	<STYLE TYPE="text/javascript">
	classes.ROJOBOLD.all.color = "red"
	classes.ROJOBOLD.all.fontWeight = "bold"
</STYLE>
</HEAD>
<BODY>
<H2 CLASS=ROJOBOLD>
	Este H2 es del tipo ROJOBOLD</H2>
<H2>Este NO</H2>
<P CLASS=ROJOBOLD>
	Este P es del tipo ROJOBOLD</P>
<P>Este NO</P>
</BODY>
```

Como vemos, se puede usar la palabra clave all para especificar que todos los TAGs dentro de la clase se ven afectados por las propiedades de la clase.

• Definiendo estilos únicos con el atributo ID

El atributo ID permite que determinadas excepciones a un estilo se cumplan. Podríamos tener un elemento que es miembro de una clase pero necesitamos que el color con el que se ve en pantalla sea diferente al de la clase:

```
<STYLE TYPE="text/javascript">
	classes.ESTILO.P.color = "red";
	classes.ESTILO.all.lineHeight = 20;
	ids.MIID.color = "green";
</STYLE>
<P CLASS="ESTILO">
	Texto ROJO Ancho.</P>
<P CLASS="ESTILO" ID="MIID">
Texto AZUL Ancho.
</P>
```

• Utilizando el método contextual()

Podemos usar el método contextual() para especificar criterios como "todos los textos STRONG de un TAG H2 serán de color rojo":

El método contextual() se utiliza para especificar criterios en nuestra hoja de estilo

```
<HEAD>
<STYLE TYPE="text/javascript">
contextual(tags.H2, tags.STRONG).color = "red"
</STYLE>
</HEAD>
<BODY>
<H2>Esto sale en color normal,
<STRONG>Esto en ROJO</STRONG>
y esto, otra vez, en color normal
</H2>
</BODY>
```

Como vemos, la selección contextual consiste en buscar un determinado TAG dentro de un contexto (otro/s TAG/s). Por ejemplo:

```
contextual(tags.P, tags.P, tags.EM).color = "red";
```

sacaría de color rojo el texto que esté dentro de un TAG <EM>, que a su vez está dentro de un TAG <P>, estando éste último dentro de otro TAG <P>. Esto es:

```
<BODY>
<P>Si usamos el anterior "contextual" yo sería
	del color por defecto.
<P>Yo también, pero
<EM> yo sería rojo</EM>
yo, de nuevo del color por defecto
</P>
y YO!</P>
</BODY>
```

• Definiendo hojas de estilo en ficheros externos

Podemos definir hojas de estilo en ficheros separados del documento y vincular las hojas de estilo al documento. La sintaxis de estos ficheros es exactamente igual a la que usamos para definir estilos en un documento HTML, sólo que los TAGs <STYLE> y </STYLE> no son necesarios. Por ejemplo:

```
// este es el fichero llamado hojas.htm
classes.ESTILO.P.color = "red";
classes.ESTILO.all.lineHeight = 20;
contextual(tags.H2, tags.STRONG).color = "red"
tags.P.color="red";
// fin del fichero
```

Para utilizar hojas.htm en nuestro documento tendremos que vincularlo, utilizando el TAG LINK, de la siguiente manera:

```
<HEAD>
<TITLE>Solo un ejemplo</TITLE>
<LINK REL=STYLESHEET
      TYPE="text/JavaScript"
      HREF="hojas.htm" TITLE="Titulo">
</HEAD>
```

• Combinando hojas de estilo

Se pueden utilizar varias hojas de estilo simultáneamente para modificar la presentación que se obtiene de nuestro documento:

```
<STYLE TYPE="text/javascript"
      SRC="uno.htm"></STYLE>
<STYLE TYPE="text/javascript"
      SRC="dos.htm"></STYLE>
<STYLE TYPE="text/javascript">
      tags.H2.color = "red";
</STYLE>
```

En el caso de que vinculemos varias hojas de estilo externas, la última tiene preferencia sobre las anteriores, esto es, si alguna de las propiedades de nuestra hoja de estilo se repite en los ficheros externos, el que se utilizará es el último incluido.

Propiedades de formato de las hojas de estilo

Las hojas de estilo JavaScript tratan a todos los elementos de bloque (los elementos que empiezan en una nueva línea como <H1>, <H2>, <P>, ...) como si estuvieran incluidos dentro de una caja. Cada caja tendrá borde, márgenes y relleno.

Las hojas de estilo JavaScript tratan a todos los elementos de bloque como si estuvieran incluidos dentro de una caja

Los márgenes indican la separación del elemento con respecto al límite del documento, mientras que el relleno o *padding* indica la distancia entre el elemento y su borde. El área de *padding* utiliza el mismo color o imagen de fondo que el elemento (con backgroundColor o backgroundImage).

El tamaño de la caja será la suma del tamaño del elemento (texto, imágenes...) más el tamaño del borde y del padding.

Definiendo los márgenes

Podemos definir los márgenes de cada elemento de bloque mediante las propiedades marginTop, marginRight, marginBottom y marginLeft. También se puede utilizar el método predefinido margins(). El siguiente ejemplo utiliza las propiedades individualmente:

```
tags.P.marginTop = 10;
tags.P.marginRight = 20;
tags.P.marginBottom = 30;
tags.P.marginLeft = 40;
```

El ejemplo anterior quedaría de la siguiente manera utilizando el método margins():

```
// Comentario; el formato de margins es:
// margins(top, right, bottom, left)
tags.P.margins(10, 20, 30, 40);
```

Si queremos que todos los elementos de bloque de nuestro documento tengan los mismos márgenes por defecto, especificaremos las propiedades de margen del TAG <BODY>. Por ejemplo:

```
tags.BODY.margins(0, 15, 10, 30);
```

Definiendo el tamaño, el color y el estilo de los bordes

El método que tendremos que emplear para definir las propiedades de los bordes en nuestros elementos de bloque es muy similar al que ya empleamos para definir los márgenes. Para definir el tamaño del borde utilizaremos las propiedades borderTopWidth, borderRightWidth, borderBottomWidth y borderLeftWidth. Al igual que para los márgenes, existe un método predefinido que nos permite definir todos los tamaños a la vez: borderWidths().

A su vez, los bordes pueden ser de dos tipos (estilos) diferentes: sólidos (planos) y 3D (con un efecto tridimensional). Para conseguir esto utilizaremos la propiedad borderStyle, asignándole los valores solid o 3D, mencionados antes. Por ejemplo:

```
tags.H3.borderStyle="solid";
tags.P.borderStyle="3D";
```

Finalmente, también podremos especificar el color del borde con la propiedad borderColor:

```
tags.H3.borderColor="green";
```

Definiendo Rellenos

También podemos definir el tamaño del relleno que rodea a un elemento de bloque, esto es, el espacio entre el elemento y los bordes. El método ya nos es conocido, podemos utilizar las cuatro propiedades de rigor (en este caso paddingTop, paddingRight, paddingBottom y paddingLeft) o utilizar el método predefinido paddings().

El fichero todo.htm (que se incluye en el CD-ROM que se distribuye con la revista) es un ejemplo a modo de resumen de las propiedades de formato de las hojas de estilo que hemos explicado anteriormente.

Observamos en el archivo todo.htm una cabecera (<H3>) y un párrafo que muestra diferentes propiedades en la hoja de estilo.

La principal utilidad de las hojas de estilo es automatizar el trabajo de los webmasters, esto es, hay tareas muy repetitivas; vemos, por ejemplo, la página del juego X-Wing vs TIE-Fighter en All Games Network, cuya dirección es http://www.allgames.com/xvt. Esta página utiliza unas pequeñas tablas para indicar a que día corresponde cada noticia (véase la imagen adjunta), lo que complica y agranda el código de la página.

Podríamos haber definido una clase para hacer exactamente lo mismo, mediante un TAG <P>, pasando el código HTML necesario para cada día con noticias de ocho líneas a una sola, lo que simplifica el mantenimiento de la página y su tamaño (cada byte es fundamental para que la página cargue mas rápido).

Tabla 1

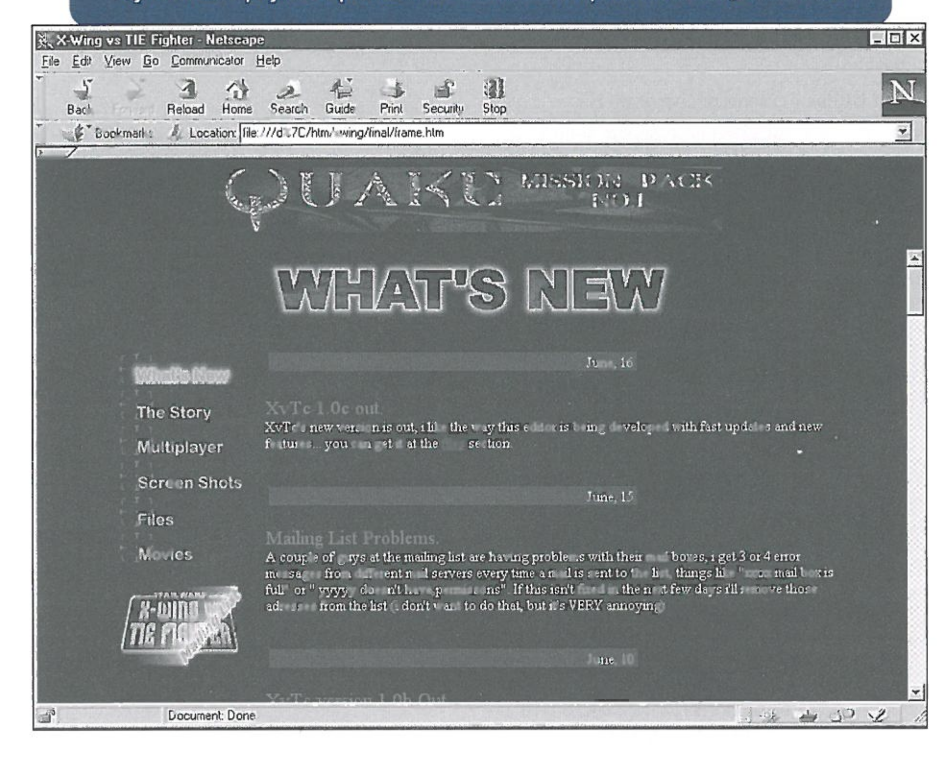
```
// Estilos y colores del borde de H3
        tags.H3.borderStyle="solid";
        tags.H3.borderColor="red";
        tags.H3.backgroundColor="green";
// Tamaños del borde de H3
        tags.H3.borderTopWidth="15";
        tags.H3.borderBottomWidth="15";
        tags.H3.borderLeftWidth="15";
        tags.H3.borderRightWidth="15";
// paddings de H3 tags.H3.paddingTop="20";
        tags.H3.paddingBottom="20";
        tags.H3.paddingLeft="100";
        tags.H3.paddingRight="100";
// Estilos y colores del borde de P
        tags.P.borderStyle="3D";
        tags.P.backgroundColor="yellow";
        tags.P.borderColor="blue";
// Tamaños del borde de P
        tags.P.borderTopWidth="10";
        tags.P.borderBottomWidth="10";
        tags.P.borderLeftWidth="10";
        tags.P.borderRightWidth="10";
// Paddings de P
        tags.P.paddingTop="10";
        tags.P.paddingBottom="10";
        tags.P.paddingLeft="10";
        tags.P.paddingRight="10";
// Márgenes de P
        tags.P.marginLeft=30;
        tags.P.marginRight=50;
        tags.P.marginTop=0;
        tags.P.marginBottom=30;
Esto es un H3 con borde sólido
Este es un párrafo con borde tridimensional.
Nótense los márgenes y los paddings, para entender mejor
cómo funcionan los elementos de una hoja de estilo
recomiendo que vayas modificando los valores
de las propiedades de la misma.
Este texto es de lo más normal.
```

Fuentes dinámicas

Otra de las interesantes novedades de Netscape Communicator 4.0 es la inclusión de las fuentes dinámicas, solución para todos nuestros males cuando queremos utilizar fuentes distintas a las estándar. Hasta ahora teníamos que utilizar fuentes más o menos estándar en nuestras páginas WEB, ya que si usábamos una fuente que el usuario no tenía en su ordenador, el browser utilizaba una fuente cualquiera, estropeando toda la presentación de la página.

Communicator permite que el usuario se baje a su ordenador la fuente que estamos usando, de igual forma que hasta ahora se bajaba una imagen o un applet Java. Cuando escribimos un documento


Figura 1. Esta página se presta como anillo al dedo para utilizar hojas de estilo.



utilizando fuentes dinámicas tendremos que especificar al principio del documento el sitio de donde Communicator bajará la fuente. Lo podemos hacer de dos maneras, mediante una hoja de estilo o con el TAG <LINK>.

Con una hoja de estilo CCS1 se haría de la siguiente forma:

```
<STYLE TYPE="text/css">
@fontdef url
(http://home.netscape.com/fonts/sample.pfr);
</STYLE>
```

Lo mismo, pero utilizando el TAG <LINK>:

```
<LINK REL=
 fontdef SRC=
 "http://www.netscape.com/fonts/sample.pfr">
```

siendo en ambos casos sample.pfr el fichero que mantiene la definición de la fuente (este fichero puede contener más de una fuente).

Una vez definida nuestra fuente, podremos utilizarla de la manera estándar, con el atributo FACE del TAG <FONT>, esto es, si el fichero de definición sample.pfr incluye una fuente llamada smashing y otra pumpkins, las utilizaríamos de la siguiente manera:

```
<FONT FACE="smashing">
Que cosas, esta fuente no la tenia</FONT>
<FONT FACE="pumpkins">
Y esta tampoco</FONT>
```

Para más información sobre fuentes dinámicas, y para bajar ficheros de definición, véanse las páginas http://home.netscape.com/comprod/products/communicator/fonts/index.html o http://www.bitstream.com/world.

Conclusiones

La lista de novedades de Communicator es grande y muy importante, Netscape sigue con su política de cantidad y calidad, con novedades que afectan tanto en el nivel del usuario como en el del programador.

Continuando con el HTML dinámico, en el siguiente número de Sólo Programadores aprenderemos a programar con *layers* (capas). Esta forma de programación constituye un gran avance de cara a conseguir importantes mejoras en la presentación visual de nuestras páginas WEB.

Bibliografía

- "JavaScript: Una chispa de vida WEB", por Fernando J. Echevarrieta. Sólo Programadores número 30.
- "El Modelo de Objetos de JavaScript", por Alejandro M. Reyero. Sólo Programadores número 33.
- "JavaScript: Frames y Windows", por Alejandro M. Reyero. Sólo Programadores número 34.
- "LiveConnect: el puente entre Java y JavaScript", por Alejandro M. Reyero. Sólo Programadores número 35.

Contactar con el autor

Si el lector quiere hacer llegar sus opiniones, preguntas sobre los artículos, ejemplos, comentarios o, simplemente, decir "hola" al autor, puede hacerlo enviando un mail a la siguiente dirección de correo electrónico:

E-Mail: axl@las.es

o bien, accediendo a la dirección siguiente:

WEB: http://www1.las.es/~axl

INTELIGENCIA ARTIFICIAL

Vamos a crear vida en un PC

Manuel de la Herrán Gascón

Nos encontramos ante un mundo habitado por hormigas artificiales capaces de cooperar o luchar entre ellas. Aprenderemos a programar de forma muy sencilla este tipo de sociedades. Discutiremos sobre cuál es el tipo de individuo con mayor probabilidad de supervivencia. Veremos cómo se produce un comportamiento emergente.

Cualquiera es capaz de reconocer un ser vivo nada más verlo. Los seres vivos nacen, comen, excretan, se relacionan unos con otros y con su entorno, tal vez se reproducen y finalmente mueren.

Tenemos la capacidad de simular en un ordenador estos comportamientos y estudiar la forma en que evoluciona una sociedad que se encuentra formada por organismos artificiales.

Las implicaciones que se derivan de todolo nateriormente expuesto pueden ser fantásticas.

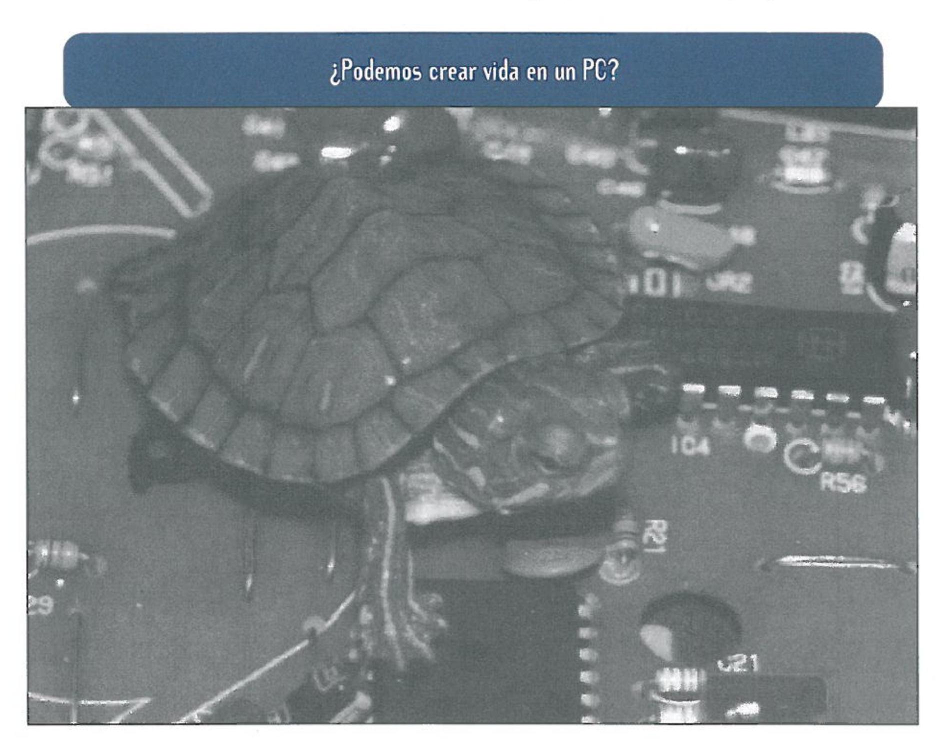
Vida Artificial

La *Vida Artificial* se puede definir como la parte de la Inteligencia Artificial que pretende reproducir los procesos y comportamientos típicos de los seres vivos.

Sin embargo, la Vida Artificial no estudia únicamente la vida tal como es, sino también tal como podría llegar a ser.

Por una parte nos encontramos con los intentos hardware de emulación de vida. Por ejemplo, es posible construir un pequeño robot con aspecto de ratón

¿Podemos crear vida en un PC?



capaz de encontrar la salida de un laberinto. Por otra parte están las simulaciones software. Éstas tienen la ventaja de que permiten construir un gran número de seres vivos y entornos en los que éstos existen, de manera que es más fácil estudiar comportamientos sociales.

La Vida Artificial nos sirve para estudiar fenómenos sociales complejos

En este artículo veremos cómo es posible programar simulaciones software de sociedades de seres vivos y obtener del estudio de las mismas sorprendentes conclusiones.

Creando vida artificial

Un programa de vida artificial normalmente está estructurado de la siguiente forma como bucle general:

```
inicializar_mundo
while detener = false

        for i = 1 to numero_de_seres_vivos
        ejecutar_ser_vivo
        next i

wend
mostrar_resultados
```

En la fase de inicialización se crea un conjunto inicial de seres vivos. Mientras no se pulse un botón de detener, se ejecutará cíclicamente cada uno de los seres.

Finalmente se muestra un resumen de la ejecución, con el procedimiento mostrar_resultados.

Hormigas y Plantas

En el CD-ROM que acompaña a esta revista se encuentra el programa de instalación y el código fuente en Visual Basic 4.0 para 16 bits de un programa de Vida Artificial.

En este programa veremos un mundo de dos dimensiones en el que conviven hormigas y plantas. Las plantas crecen cuando son regadas. Las hormigas pueden comer plantas, moverse, regar plantas, pelearse y reproducirse.

Las hormigas ganan energía cada vez que comen una planta. Cuando una planta ha sido suficientemente regada, será comida por la primera hormiga que pase junto a ella. No hay límite en la cantidad de plantas que una hormiga puede comer.

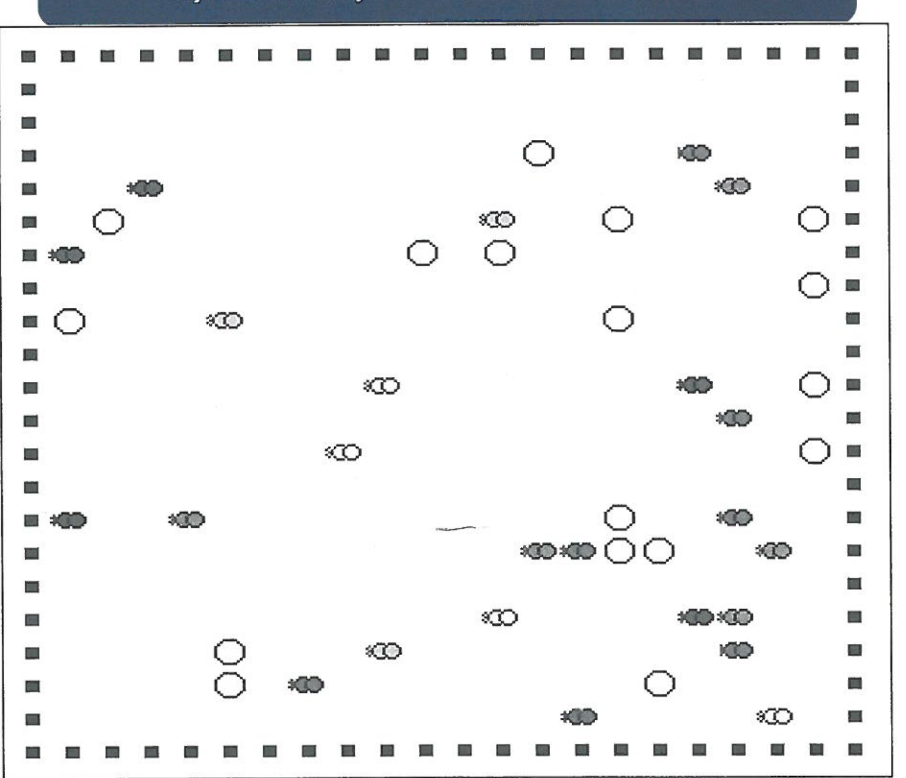
Las hormigas pierden energía cada vez que se mueven, riegan, se pelean o se reproducen. Si una hormiga pierde toda su energía, muere. Por otra parte, cada cierto tiempo nace una planta de forma espontánea.

La Vida Artificial estudia la vida tal como podría llegar a ser

En un combate, la hormiga ganadora se quedará con la mitad de la energía (comida) acumulada por la perdedora. Además, cuantas más veces salga vencedora una hormiga, más probabilidades tendrá de salir victoriosa de la siguiente pelea.

La ejecución que sigue el proceso que imita el comportamiento de una hormiga se puede descomponer en las acciones que detallamos a continuación:

Figura 2. Las hormigas se mueven al azar buscando comida.



Ejecutar ser vivo

```
if comida_acumulada = 0 then
        muerte_de_ser_vivo
else
comprobar_vecinos
comer_plantas
accion =decidir_accion_a_realizar
select case accion
        case 1
                movimiento
        case 2
                regar_planta
        case 3
                reproducirse
        case 4
                pelearse
end selecto
end if
```

En primer lugar, si la hormiga no posee comida, la declaramos muerta. En caso contrario, la hormiga observa si existen plantas u otras hormigas a su alrededor. Si existen plantas comestibles, se las come. Después, podrá regar las plantas que tdavía no son comestibles, pelear o reproducirse con sus vecinos o, simplemente, seguir paseando por su mundo virtual.

Los procesos regulares descritos se dan en la naturaleza entre especies que son depredador y presa y en la sociedad, con la oferta y la demanda

En este programa nos encontramos con cinco tipos de hormigas distintas. La probabilidad de que una hormiga realice una determinada acción dependerá del tipo de hormiga de que se trate y de los individuos con que se encuentre.

Cuando dos hormigas se reproducen, se crea una nueva hormiga con características similares a las de los padres. Por ejemplo, una hormiga de tipo 1 (roja) y una hormiga de tipo 5 (verde) generarán con gran probabilidad un descendiente de tipo 3 (naranja). Además, para que la sociedad no se estanque en un único tipo de hormiga, se permitirá, por ejemplo, que dos hormigas verdes produzcan una roja; eso sí, con una probabilidad mucho más baja.

La figura 1, en la que podemos observar los tipos de hormigas, sus comportamientos y la probabilidad asociada a cada uno, tiene una interpretación muy sencilla. La hormiga roja es la más egoísta. Nunca riega y cuando se encuentra con otra hormiga se dedica a pelear o a reproducirse. La hormiga verde es la menos egoísta, no lucha nunca y se dedica a regar todas las plantas que encuentra a su paso. El resto de hormigas son términos medios entre estos dos extremos.

Es evidente que las hormigas del tipo verde son la base sobre la que se sustenta la sociedad. Es decir, si no hay quien riegue las plantas, éstas no crecen, la comida se termina y las hormigas mueren.

Haciendo pruebas sobre las simulaciones con distintos valores nos damos cuenta de que hay una constante que siempre se produce. Basta con ejecutar el programa dos o tres veces para verlo. ¿Qué será?

Comportamientos cíclicos

En el mundo artificial de hormigas y plantas, la acción de "regar" es un acto que repercute en beneficio de la población completa, ya que cuando una hormiga riega una planta no se asegura que sea ella misma quien la coma después.

Una sociedad formada únicamente por hormigas rojas moriría rápidamente

Figura 1. Tipos de hormigas.

Tipo de Hormiga	¿Hay Planta?	¿Hay Hormiga?	Mover	Regar	Luchar	Reproducirse
Roja	No	No	100%	0%	0%	0%
	No	Si	0%	0%	80%	20%
	Si	No	100%	0%	0%	0%
	Si	Si	0%	0%	80%	20%
Rosa	No	No	100%	0%	0%	0%
	No	Si	20%	0%	60%	20%
	Si	No	75%	25%	0%	0%
	Si	Si	20%	20%	40%	20%
Naranja	No	No	100%	0%	0%	0%
	No	Si	40%	0%	40%	20%
	Si	No	50%	50%	0%	0%
	Si	Si	20%	30%	30%	20%
Amarilla	No	No	100%	0%	0%	0%
	No	Si	60%	0%	20%	20%
	Si	No	25%	75%	0%	0%
	Si	Si	20%	40%	20%	20%
Verde	No	No	100%	0%	0%	0%
	No	Si	80%	0%	0%	20%
	Si	No	0%	100%	0%	0%
	Si	Si	0%	80%	0%	20%

de hambre, ya que ninguna regaría las plantas. Una sociedad formada únicamente por hormigas verdes disfrutaría de gran prosperidad.

Pero en nuestra simulación siempre habrá hormigas de todos los tipos. Podrían aparecer hormigas rojas voraces y casi invencibles. Sin embargo, las hormigas rojas dependen de las demás hormigas, ya que son por definición incapaces de regar.

Existe un comportamiento cíclico en cuanto al número de hormigas de cada tipo y también entre el número total de hormigas y el número de plantas. Si en un primer momento hay gran cantidad de plantas, más tarde será alto el número de hormigas, ya que éstas podrán regar y comer, y alcanzar el nivel de energía necesario para reproducirse.

Pero al haber muchas hormigas, las plantas desaparecen devoradas por ellas. Más adelante, al haber muchas hormigas y pocas plantas, no existirá comida para todas y el número de hormigas decrecerá. Una vez que hemos llegado al punto en el que la situación es que existen pocas hormigas y pocas plantas, comenzará el aumento del número de plantas, repitiéndose, de esta forma, el ciclo.

Las hormigas artificiales son una analogía de ciertos procesos económicos

Estos procesos regulares se dan en la naturaleza entre especies que son depredador y presa, y también en nuestra sociedad, en los conceptos de oferta y demanda. Por ejemplo, cuando cierto año la cantidad cosechada de un producto agrícola es muy pequeña, el precio sube, ya que el producto es difícil de conseguir. Este aumento del precio puede animar a los agricultores, que han observado la rentabilidad de este producto y por tanto el año siguiente se cultiva en mayor cantidad. El resultado es un exceso de producción que provoca que el precio decrezca. En este momento, el interés por cultivar este producto disminuye y nuevamente será escaso en el siguiente año, repitiéndose de esta forma el ciclo.

El dilema del prisionero

La siguiente variación del conocido *dilema del prisionero* ayudará a explicar las posibilidades de la Vida Artificial en la simulación y resolución de problemas reales.

Supongamos un conjunto de individuos; cada uno de los cuales posee una determinada cantidad de dinero. Cada persona puede decidir si invertir ese dinero en un cierto negocio y, en el caso de hacerlo, qué cantidad invertir.

El negocio en cuestión es tan fabuloso que duplica inmediatamente cualquier inversión, pero con la peculiaridad de que el total del dinero resultante (el doble del dinero invertido) es repartido por igual entre todos los individuos, independientemente de la cantidad entregada por cada uno.

Veamos los casos extremos. Por ejemplo, si existen diez individuos y todos invierten una peseta, la inversión de diez pesetas producirá otras diez de beneficio y el resultado de veinte pesetas. será repartido entre todos, con lo que cada individuo obtendrá entonces dos pesetas.

Si de los diez individuos nadie invierte, no hay beneficio y todos se mantienen como estaban. Pero si sólo uno de ellos invierte una peseta, la peseta se multiplica por dos y estas dos pesetas serán repartidas entre los diez, y ahora todos tendrán 1,2 pesetas excepto el desprevenido inversor que tendrá 0,2 pts.

Si de los diez individuos, sólo cinco de ellos invierten la peseta, el resultado de diez pesetas será repartido entre los diez, de manera que los que no invirtieron ganarán una peseta y los que invirtieron se quedarán tal como estaban al principio.

Figura 2. El dilema del prisionero.

Individuo	Cantidad Inicial	Cantidad Invertida	Cantidad Final
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	0	2
7	1	0	2
8	1	0	2
9	1	0	2
10	1	0	2
Total	10	5	15

Cada individuo desconoce siempre las inversiones que harán el resto de individuos. El objetivo es obtener el máximo beneficio posible, pero cada individuo se encuentra entonces con un gran dilema: si confía en la bondad de sus compañeros y supone que éstos realizarán grandes inversiones, debería invertir la mínima cantidad posible para obtener así los máximos beneficios sin arriesgar su dinero. Sin embargo, los demás no son tontos, y serán capaces de llegar a la misma conclusión. En ese caso, si todos invierten la mínima cantidad, el beneficio será el mínimo para todos y el negocio tan fabuloso que teníamos no estará siendo de utilidad para nadie.

Por tanto, lo más inteligente parece colaborar e invertir todos la máxima cantidad para obtener todos el máximo beneficio. Pero, ¿puede uno confiar en que los demás también lo harán?

El individuo produce de esta forma un razonamiento circular que se debate entre el beneficio propio y el beneficio que obtiene el grupo al que pertenece.

En los sistemas complejos podemos decir que 1 + 1 = 4

Se puede pensar que éste es un caso artificioso, sin paralelismo en la vida real. Todos sabemos que no existen en los mercados de valores inversiones que dupliquen su valor inmediatamente. Sin embargo, sí existe en nuestro mundo una mercancía cuyo valor se multiplica cuando se suma. Es decir, existe algo para lo que podemos afirmar que 1 + 1 = 4. Algo en lo que *"El todo es más que la suma de las partes"*.

Ese "algo" misterioso es el conocimiento. El conocimiento es el "dinero" que las personas pueden "invertir" y obtener multiplicado. Una persona puede plasmar en un documento conocimientos que le ha costado meses descubrir. Otra persona podrá en tan solo unas horas aprender sobre ese tema. Bastará con leer el documento. ¿Que sería de los programadores sin los manuales? Ahí los tenemos, por millares, a disposición de todos e independientemente de si se ha participado en su redacción.

Una vez metidos en harina, el dilema del prisionero se puede observar en un imaginario equipo de desarrollo informático en el que algunos puestos de trabajo tal vez desaparezcan si el proyecto fracasa. Los programadores pueden optar por ocultar su conocimiento al resto, con la intención de destacar y asegurar su permanencia en la empresa. Pero si todos se comportasen de esta forma empezarían los problemas: tareas que se repiten innecesariamente, errores cometidos una y otra vez,... el proyecto se convierte en un caos y tal vez nadie, en estas condiciones, conserve su puesto.

No existe dilema cuando el beneficio obtenido en la cooperación es muy alto y el perjuicio muy bajo o nulo

Si todos compartiesen todo su conocimiento, el proyecto tendrían más probabilidades de éxito y sería más probable que se mantuvieran todos los puestos de trabajo. Pero ¿se puede confiar en que los demás también lo harán?

No existe *dilema* cuando el beneficio obtenido en la cooperación es muy alto y el perjuicio muy bajo o nulo. Supongamos que el coche de un individuo queda atascado en el barro. Él sin ayuda no puede

Así puede moverse la hormiga

Procedimiento s_mover_hormiga

```
Sub s_mover_hormiga(hor As Integer, direccion As Integer)

  'Borro la hormiga
  s_pintar_objeto hormiga_y(hor), hormiga_x(hor), "circulo", "blanco"

  Select Case direccion
    Case 1
      hormiga_x(hor) = hormiga_x(hor) + 1
      hormiga_y(hor) = hormiga_y(hor) + 1
    Case 2
      hormiga_x(hor) = hormiga_x(hor)
      hormiga_y(hor) = hormiga_y(hor) + 1
    Case 3
      hormiga_x(hor) = hormiga_x(hor) - 1
      hormiga_y(hor) = hormiga_y(hor) + 1
    Case 4
      hormiga_x(hor) = hormiga_x(hor) + 1
      hormiga_y(hor) = hormiga_y(hor)
    Case 5
      hormiga_x(hor) = hormiga_x(hor) - 1
      hormiga_y(hor) = hormiga_y(hor)
    Case 6
      hormiga_x(hor) = hormiga_x(hor) + 1
      hormiga_y(hor) = hormiga_y(hor) - 1
    Case 7
      hormiga_x(hor) = hormiga_x(hor)
      hormiga_y(hor) = hormiga_y(hor) - 1
    Case 8
      hormiga_x(hor) = hormiga_x(hor) - 1
      hormiga_y(hor) = hormiga_y(hor) - 1
  End Select
  If hormiga_y(hor) = 0
Then hormiga_y(hor) =
numero_de_celdas_eje_y
  If hormiga_y(hor) =
numero_de_celdas_eje_y + 1
Then hormiga_y(hor) = 1

  If hormiga_x(hor) = 0
Then hormiga_x(hor) =
numero_de_celdas_eje_x
  If hormiga_x(hor) =
numero_de_celdas_eje_x + 1
Then hormiga_x(hor) = 1

  s_pintar_objeto hormiga_y(hor), hormiga_x(hor), "hormiga"
& hormiga_tipo(hor), "rojo"

End Sub
```

Procedimiento s_inicializar en Visual Basic. El código completo se encuentra en el CD-ROM.

```
Sub s_inicializar()

   Dim i As Integer
   Dim j As Integer
   Dim x As Integer
   Dim y As Integer
   Dim prueba As Integer

      'Totales de hormigas que se han creado hasta el momento
   numero_de_hormigas_creadas = 0
   For i = 1 To 5
      numero_de_hormigas_creadas_de_tipo(i) = 0
   Next i
   'Total de plantas que se han creado hasta el momento
   numero_de_plantas_creadas = 0

   'Hormigas actuales
   If numero_total_de_hormigas > 0 Then
      ReDim hormiga_x(1 To numero_total_de_hormigas)
As Integer
      ReDim hormiga_y(1 To numero_total_de_hormigas)
As Integer
      ReDim hormiga_tipo(1 To numero_total_de_hormigas)
As Integer
      ReDim hormiga_energia(1 To numero_total_de_hormigas)
As Long
      ReDim hormiga_accion_anterior(1 To numero_total_de_hormigas)
As String
      ReDim hormiga_probabilidad_ganar(1 To numero_total_de_hor-
migas)
As Double
   End If
   For i = 1 To numero_total_de_hormigas
      hormiga_x(i) = 0
      hormiga_y(i) = 0
      hormiga_tipo(i) = 0
      hormiga_energia(i) = 0
      hormiga_accion_anterior(i) = "nada"
      hormiga_probabilidad_ganar(i) = 50
   Next i

   'Plantas actuales
   If numero_total_de_plantas > 0 Then
      ReDim planta_x(1 To numero_total_de_plantas)
As Integer
      ReDim planta_y(1 To numero_total_de_plantas)
As Integer
      ReDim planta_agua(1 To numero_total_de_plantas)
As Integer
   End If
   For i = 1 To numero_total_de_plantas
      planta_x(i) = 0
      planta_y(i) = 0
      planta_agua(i) = 0
   Next i

   detener = False
   FRM_ej1.Comenzar.Enabled = False
   FRM_ej1.terminar.Enabled = True

   FRM_ej1.Refresh
   FRM_ej1.Frame1.Visible = False
   FRM_ej1.Frame2.Visible = False

   s_borrar_todo

   s_crear_plantas_iniciales
   s_crear_hormigas_iniciales

End Sub
```

moverlo, pero con la ayuda de otra persona sí podría. El beneficio que obtiene uno de ellos en la cooperación es muy alto mientras que el perjuicio del otro es muy pequeño, apenas un ligero esfuerzo. En estos casos, la tendencia más habitual será a la cooperación.

Existen otros ejemplos. Pensemos en la cantidad de personas que se ven involucradas en el proceso que desemboca en que podamos tomarnos una tostada con mermelada por las mañanas. Unos intervinieron en la obtención de harina, otros condujeron los camiones en los que se transporta, otros construyeron las carreteras por las que van los camiones, etc. Nos damos cuenta de que las personas estamos todas relacionadas de alguna forma. Es fácil comprender que la especialización en el trabajo es beneficiosa para la sociedad y produce un mejor aprovechamiento de los recursos.

Sin embargo, no es necesario que una entidad superior organice este reparto de tareas. La organización se produce de forma automática, es un proceso que emerge de la interacción de los objetivos de cada individuo. Es suficiente con que que las entidades actúen en su propio beneficio para que se produzca una organización adecuada.

Por ejemplo, si hay escasez de informáticos, esta situación animará a los estudiantes, que llenarán las aulas para obtener los conocimientos necesarios y esa demanda quedará atendida.

¿Sirve para algo la Vida Artificial?

El cerebro humano está formado por neuronas que trabajan simultáneamente, pero nuestro razonamiento consciente es secuencial. Por eso es tan difícil para nosotros comprender el comportamiento de un sistema de elementos muy interrelacionados. Pero para un ordenador, resulta mucho más fácil.

La Vida Artificial es un mecanismo para la representación de sistemas complejos. Una vez representado un sistema, podemos experimentar con él en situaciones hipotéticas.

En los sistemas complejos una acción simple puede propagarse

En un modelo siempre existe simplificación, ya que lo contrario sería realizar una copia idéntica, átomo por átomo. Se trata de no tener que construir una central nuclear para experimentar y prever comportamientos en una central nuclear.

Por ejemplo, si construimos un modelo de una fábrica, podemos simular posibles inversiones como compra de nueva maquinaria o ampliación de plantilla y comprobar si los resultados obtenidos son los deseados.

Complejidad y comportamiento emergente

En los sistemas complejos puede producirse lo que se denomina el *comportamiento emergente*. La gran ventaja de la Vida Artificial es que permite observar fenómenos emergentes. Las propiedades emergentes son aquéllas que aparecen en los sistemas como resultado de la interacción entre sus partes y que no pueden explicarse a partir de las propiedades de los elementos que lo componen.

En estos sistemas, una acción simple se puede propagar exponencialmente provocando efectos de gran envergadura. Por esta razón se dice que estos sistemas se encuentran en la frontera entre el orden y el desorden.

Las investigaciones llevadas a cabo en este campo se han realizado en fechas bastante recientes y por lo tanto llegarán a un nivel de profundidad mucho mayor con el paso del tiempo. La complejidad es un fenómeno complejo ;-), y describir ejemplos simples de complejidad resulta, en efecto, una paradoja. Sin embargo, eso es precisamente lo que vamos a intentar hacer a continuación, por medio de algunos ejemplos sacados de nuestra experiencia.

El primer caso: un montón de arena sobre el que dejamos caer granos lentamente. En un momento dado sucede que, a partir de cierto ángulo, la caída de un sólo grano sobre el montón ya formado provocará avalanchas de gran tamaño. De la misma manera, la caída de un copo de nieve, inocente e inofensivo en sí mismo, puede provocar la ruptura de la rama de una árbol sobre la que se ha acumulado gran cantidad de copos.

Una colonia de hormigas naturales puede llevar a cabo tareas de gran complejidad, como explorar el entorno, construir galerías muy complicadas o decidir si es adecuado o no un tipo de alimento, con la lamentable pero ineludible pérdida de algunas hormigas en el proceso. Pero las hormigas, si son consideradas una por una, son totalmente incapaces de realizar ningunas de estas tareas. El comportamiento del hormiguero emerge a partir de interacciones entre elementos simples (hormigas) y no se puede explicar únicamente si nos basamos en las propiedades que tiene un individuo en concreto. En este caso, el hormiguero podría ser considerado como un ser vivo de un nivel superior a la hormiga.

Conclusión

La vida artificial es una metodología que nos permite representar en un ordenador modelos simplificados de mundos imaginarios, de nosotros mismos o del mundo que nos rodea. En el artículo que nos ocupa, hemos utilizado la Vida Artificial para analizar determinados comportamientos sociales.

Las propiedades emergentes son resultado de la interacción entre las partes de un sistema y no pueden explicarse a partir de las propiedades de los elementos que lo componen

En los próximos números estudiaremos los *Algoritmos Genéticos*, con los cuales es posible resolver problemas utilizando únicamente una pequeña parte de la Vida Artificial: la reproducción.

También se discutirá sobre la forma de proporcionar a los seres vivos artificiales la capacidad de aprender.

En el CD-ROM que acompaña a la revista se incluyen los códigos fuente completos en Visual Basic referentes a los programas mencionados en el artículo.

¿TE ATREVES?

Desde **Sólo Programadores** estamos preparándonos para un torneo muy especial, de ordenador a ordenador, de programador a programador. El torneo de ajedrez está empezando a tomar forma y éstas son las bases, aunque también podéis informaros en AWS en la calle Serrano Jover, 3, de Madrid, o en el teléfono (91) 5 42 50 87. Para conocer cuál será el alcance de la participación, os pedimos que os inscribáis enviando vuestros datos a la dirección de AWS o a la dirección de e-mail: **solop@towercom.es**. El periodo de inscripción ya ha comenzado y terminará el 30 de noviembre, y la fecha límite para la recepción de los programas es el 15 de diciembre de 1997. La celebración del torneo será anunciada con suficiente antelación y podréis participar en persona o desde vuestra propia ciudad, ya que vuestro programa os representará y en todo momento estará presente un notario.

BASES DEL TORNEO

1 Se presentará un programa que juegue al ajedrez recibiendo la jugada del contrario y devolviendo la propia. El tiempo de la partida será medido por un juez y la partida se jugará físicamente en una mesa con un tablero siguiendo las instrucciones que cada ordenador dé; cada ordenador avisará de que ha elegido un movimiento con un pitido y mostrará su jugada en la pantalla de la forma: casilla a mover - casilla destino. La notación de la casilla vendrá dada por su columna y su fila. Para las columnas se utilizarán las de la A a la H siendo la columna A la primera columna de la izquierda, visto el tablero desde la posición de las blancas. Para las filas utilizaremos los números del 1 al 8 siendo la fila 1 la más cercana a nosotros (visto el tablero desde el lado de las blancas). Como ejemplo, el movimiento del peón de rey dos lugares hacia adelante sería: "E2 - E4". La jugada que hace el contrario se anotará en el ordenador de forma manual tecleando "E2-E4" <RETURN>.

2. El programa jugará al ajedrez según las normas de este juego, admitiendo todos sus movimientos.
3. El programa se ejecutará bajo Windows 95, MS Dos 6.22 y Linux, en un Pentium 166 con 16 MB de RAM.
4. El programa podrá jugar con las blancas o con las negras indistintamente.
5. Se debe entregar el código fuente, la versión del compilador utilizado y una breve documentación para su utilización.
6. Un programa podrá ser descalificado si realiza un movimiento no válido y esto deberá ser advertido por el contrario; por lo tanto, un movimiento será válido siempre que el otro programa no nos advierta de lo contrario. De la misma forma, si un programa nos advierte de un movimiento erróneo y éste no lo es, será descalificado.
7. Los jueces son los encargados de advertir las posiciones de tablas, el programa no advertirá de ello.
8. El programa debe detectar si ha ganado la partida y quedará descalificado ante cualquier detección errónea. No se avisará al contrario en el caso de que deje al rey al descubierto y ganará la partida el que antes mate al rey.
9. La metodología del concurso se reservará hasta que sea conocido el número de participantes (eliminatoria, puntos, etc.)
10. No se podrán utilizar en el programa bases de datos adicionales.
11. Los derechos del programa ganador pasarán a la propiedad de la empresa Tower Communications, SRL.
12. El premio consistirá en un Pentium PRO 200 con 64 MB RAM, VIRGE4 MB, CDX 16, disco duro 5 GB, monitor 17", teclado, ratón y disquetera.

CONCURSO DE PROGRAMACIÓN

SÓLO PROGRAMADORES Y AWS INFORMATICA

SERRANO JOVER 3, MADRID

MÁS DATOS EN EL PRÓXIMO NÚMERO

Análisis del Pentium II

Jorge Figueroa

Hacia el mes de febrero del presente año, se presentó en la International Solid State Circuits Conference de San Francisco lo que era el primer prototipo operativo del por entonces conocido aún con el nombre temporal de Klamath.

Este primer prototipo destacó sobre todo porque funcionaba a una velocidad interna de 433 Mhz y ello produjo un gran impacto entre los expertos del sector, que no esperaban tal velocidad de proceso. Poco tiempo después, el 7 de mayo, se presentó al mercado la versión comercial de dicho procesador con el nombre comercial de Pentium II y en un brevísimo espacio de tiempo, hacia el mes de Junio, ya se podían encontrar algunos modelos de ordenadores de gama alta de diversas marcas con dicho microprocesador incorporado.

Principales características del Pentium II

El Pentium II, a pesar de que ha sido diseñado tomando como base el núcleo del conocido Pentium Pro (del cual ha copiado hasta un bug), posee algunas diferencias y novedades bastante significativas, que a continuación pasamos a detallar:

• Diseño:

Este procesador es el primero de la familia Intel que se presenta en una caja diferente, pasando de estar en un encapsulado típico plano tipo PGA, a estar soldado junto a la memoria caché level 2, en una tarjeta llamada SEC que se conecta a un slot llamado SLOT ONE mediante las conexiones que posee para ello en uno de los lados (como una tarjeta de expansión típica en lo que a método se refiere).

• Tecnología de fabricación:

Está formado por un total de 7,5 Millones de transistores (2 millones más que el Pentium Pro), se fabrica con una escala de integración de 0,35 micras (casi la mitad que el P6) y está basado en tecnología CMOS.

• Microarquitectura:

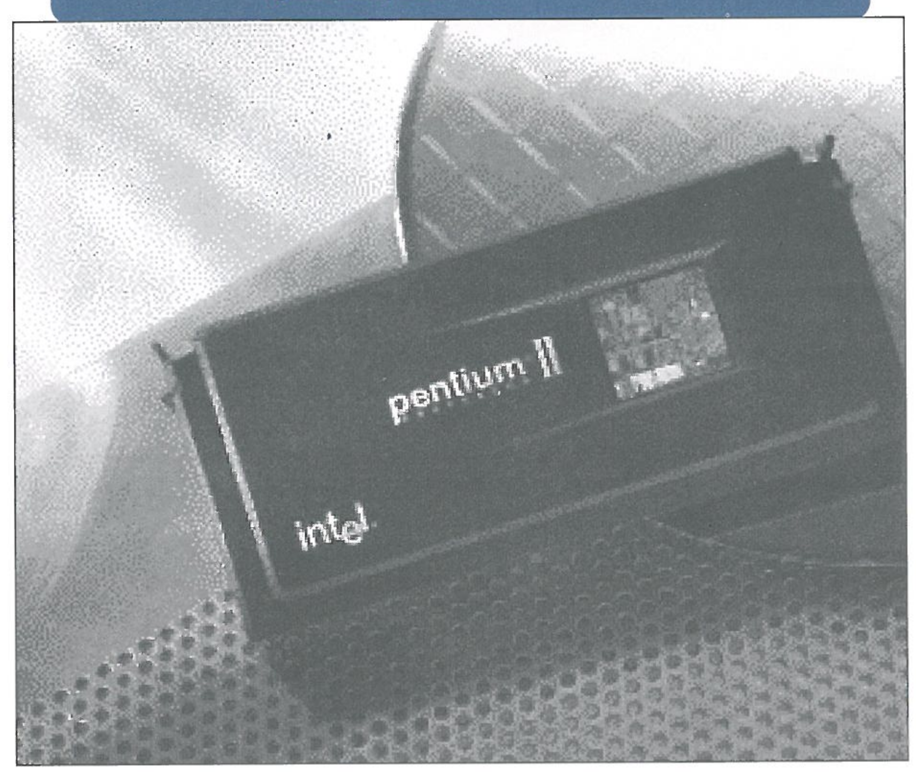
El Pentium II está formado también por un total de cinco unidades de procesamiento paralelo, siendo una para mediar con la memoria, dos para enteros y dos para reales (coprocesador) que forman parte de las doce etapas de procesado en que está diseñado el micro.

Por otra parte, se basa en la llamada tecnología *Dynamic execution* que se explicó a fondo en el articulo sobre el Pentium Pro (ver Sólo Programadores nº

El Pentium II parece estar destinado a convertirse en el procesador que sustituirá tarde o temprano a toda la familia Pentium. En este artículo se van a detallar todos los datos que destacan como novedad de esta nueva generación de micros.



Figura 1. Encapsulado SEC de un Pentium II.



35). Recordaremos sólo que básicamente consiste en la combinación de tres tecnologías que hacen que el rendimiento global de ejecución sea mucho mayor que en un procesador convencional de ejecución totalmente secuencial.

Pentium II funciona ya a velocidades de hasta 300 Mhz

• Velocidad:

Los tres primeros modelos que se han presentado funcionan a una velocidad de 233, 266 y 300 Megahertzios, lo que significa un avance respecto a los anteriores micros, que hasta ahora nunca habían superado la frontera de los 200 Mhz.

Dado que el primer prototipo que se mostró antes de su presentación oficial en el mercado funcionaba a más de 400 Mhz, es de esperar que en un futuro próximo saldrán versiones más potentes a medida que los modelos actuales puedan bajar de precio y se extiendan más en el mercado de consumo.

• Memoria caché:

El Pentium II incorpora, como los nuevos Pentium MMX, un total de 32 Kbytes de memoria caché level 1, la cual está dividida en dos sub-bloques de 16 Kbytes destinados a código y datos respectivamente.

La memoria caché level 2 del Pentium II es de 512 Kbytes del tipo unificado. Como ya pasaba en el Pentium Pro, está integrada en la propia tarjeta SEC (el encapsulado) donde se encuentra el procesador y está conectada a la CPU con un bus de datos que funciona a la mitad de velocidad del micro. Para más información sobre este curioso dato, ver el apartado que se encuentra más adelante sobre el tema.

• Instrucciones/funcionalidad:

Este micro es totalmente compatible con todos los micros de la familia x86 (o sea, puede ejecutar todo el soft existente en el mercado) y, además de incluir todas las mejoras del Pentium Pro, incorpora las extensiones MMX que antes sólo estaban disponibles en los Pentium MMX.

También se ha optimizado para funcionar de forma más rápida en modo real, lo que hace que las aplicaciones de 16 bits se ejecuten más deprisa que en un Pentium normal, y no sólo las de 32 bits, como pasaba en el Pentium Pro, lo que hará que el mercado doméstico, que está aún muy saturado por el soft de híbrido (como es Win95) con código tanto en 16 como 32 bits, se beneficie también de un mayor rendimiento de ejecución.

Su principal novedad respecto al Pentium Pro es que incorpora también las extensiones MMX

Para más información sobre las extensiones MMX el lector puede leer los dos artículos aparecidos sobre el tema en el número 33 de esta misma publicación que aparecieron bajo los títulos de "La tecnología MMX" y "MMX, Hardware y Programación" (páginas 52 y 59).

Resumen de otras características

Este micro posee otras características secundarias que son también dignas de mención:

El Pentium II está diseñado para poder ser instalado en sistemas tipo

biprocesador, como podrían ser estaciones gráficas y similares.

El chip posee sistemas integrados para la seguridad de los datos que circulan por el bus, incluyendo soporte para memoria ECC, análisis de errores, recuperación de bits en caso de error y chequeo de redundancia. En total el sistema puede detectar y corregir fallos de hasta dos bits de datos en el bus, lo cual será de agradecer en sistemas de datos que gestionen un gran volumen de datos.

Rendimiento entre x1.6 y x2.0 veces superior a un Pentium de 200 Mhz típico. Un procesador Pentium II a 266 Mhz ofrece un ICOMP equivalente a un Pentium MMX de unos 350 Mhz.

Incorpora la Arquitectura *Dynamic Execution,* que se resume en la capacidad del micro para ejecutar código de forma "especulativa" y no secuencial.

Gracias al encapsulado SEC, se hace más fácil y flexible el diseño de placas base.

Está diseñado con dos buses separados, uno para caché y otro para memoria, lo que hace que el movimiento de datos sea mucho más fluido y que se produzcan, en consecuencia, menos cuellos de botella.

El Pentium II es el primer chip Intel que usa el sistema SEC de encapsulado

Posee la Unidad de punto flotante (FPU) optimizada con capacidad de soportar hasta 300 Millones de Instrucciones Por Segundo (MIPS) a 300 Mhz de velocidad de procesado (una instrucción por ciclo).

Posee un sistema integrado de autotesteo del hardware, con capacidad para examinar fallos de microcódigos, verificar la caché de datos, la caché de instrucciones e incluso la ROM. Todo ello se suma a las capacidades de control que incluía ya el Pentium Pro para el control de eventos de diversas índoles (ver artículo sobre Pentium Pro para más detalles) y que permitían monitorizar muchos aspectos del funcionamiento interno del micro gracias a una serie de registros contadores que incluía.

Mercado asignado

Como sucede siempre cuando aparece un modelo nuevo de procesador, esta última generación de chips será la que se incorporará a partir de ahora en los nuevos ordenadores de gama alta que aparezcan en el mercado, tanto en servidores como en estaciones gráficas y matemáticas, dado que su precio actual es muy elevado para el sector doméstico (unas 150.000 pesetas un chip Pentium II a 266 Mhz) y el propio chip tiene casi el precio de un ordenador completo de gama media.

Tanto el Pentium Pro como el Pentium II tienen un BUG en la FPU

La ventaja derivada de todo ello es que los modelos inferiores bajarán más de precio y que los usuarios domésticos podrán adquirir sistemas más potentes por el mismo precio, con lo que el nivel de los programas comerciales podrá aumentar en cuanto a los requerimientos y las prestaciones se refiere.

Fallos detectados

Pese a lo reciente de la presentación de este micro, hay ya reconocido oficialmente por la propia Intel (casi desde que

Figura 2. Test de comparación entre Pentium II y modelos anteriores.

RESULTADOS COMPARACIÓN	Pentium 200 Mhz	Pentium MMX 200 Mhz	Pentium Pro 200 Mhz 256 Kb L2	Pentium II 233 Mhz 512 Kb L2	Pentium II 266 Mhz 512 Kb L2
SPEC CPU95* (UNIX*)					
SPECint_Base95*	5.00	6.41	8.20	9.49	10.80
SPECfp_Base95*	2.98	3.90	5.54	5.91	6.43
WINDOWS					
Norton SI-32* Windows 95	43.8	56.7	90.0	112.6	123.7
CPUmark32* Windows 95	382	423	553	606	693
Intel Media Benchmark	153.06	253.08	196.29	310.25	351.10
Vídeo	153.42	267.23	160.97	269.48	304.40
Procesado de imagen	157.77	742.65	222.04	1022.94	1129.15
Geometría 3D	155.69	160.19	212.41	246.74	280.22
Audio	148.50	323.81	239.27	403.03	457.78

apareció) un bug de microcódigo que produce fallos numéricos bastante serios.

Este bug (error), que existe también en su predecesor, el Pentium Pro, consiste en un error de funcionamiento que posee el coprocesador matemático en el sistema de detección de errores de conversión de datos. Pasamos a detallarlo a continuación. Los coprocesadores matemáticos pueden operar con varios tipos de datos incluyendo tanto enteros de 16, 32 y 64 bits, como datos en punto flotante de también varios tamaños, aunque internamente el coprocesador sólo puede operar con datos en punto flotante de 80 bits (tipo de dato capaz de albergar a todos los demás tipos inferiores mencionados).

Dada esta limitación, creada expresamente para poder simplificar así enormemente la arquitectura interna del procesador numérico, cada vez que se introduce un dato, sea del tipo que sea, éste se convierte de forma automática y transparente al programador en un real de 80 bits. Posteriormente, cuando el dato o un resultado se extrae de nuevo, simplemente se reajusta su tamaño y/o tipo.

Este micro se puede encontrar ya en servidores de gama alta

Hasta aquí todo es normal. El problema aparece cuando se extrae un dato contenido en un registro de la FPU de 80 bits y se convierte a un tipo y/o longitud inferior (por ejemplo un entero de 32 bits), porque pese a que las conversiones las realiza bien, cuando ocurre que el dato no cabe por las limitaciones de rango en el tipo de dato de destino, el microprocesador dispone por diseño (como indican sus manuales de programación) de un sistema gracias al cual este error (evento) es detectado y uno de los bits (una bandera) de uno de los registros de estado del coprocesador se pone a 1 para indicar (dejar

Tabla 1. Fragmento del código que testea el bug del Pentium II/Pro. La versión completa aparece en el CD-ROM.

```
; Tabla 1, PROGRAMA <FISTBUG> .
;  Este programa testea el Bug del Pentium Pro/II de la FPU.
;  Copyright (c) 1997-Present Robert Collins
;  Robert R. Collins         email: rcollins@x86.org
;------------------------------------------------------------------------
      .xlist                              ; disable list file
      .286
      FSW_IE   equ       1                        ; Status Word IE bit
PRINT_PASS_FAIL MACRO
ifdef VERBOSE
      pushf                                       ; save results of comparison
      mov        ah,9                             ; print string function ID
      mov        dx,offset PassMsg       ; prepare for test passed
      jnz        @F                               ; test did pass
      mov        dx,offset FailMsg       ; get address of failed message
@@:          int          21h                          ; print message
      popf                                        ; restore flags
endif                                                  ; VERBOSE
ENDM
PRINT_MSG  MACRO  MSG
ifdef VERBOSE
      mov        ah,9
      mov        dx,offset MSG
      int        21h
endif                                                  ; VERBOSE
ENDM
; 16-bit Floating point environment
;------------------------------------------------------------------------
  FPU_Struct   STRUCT
      FCW                   dw       ?        ; Control word
      FSW                   dw       ?        ; Status word
      FTW                   dw       ?        ; Tag word
                            dw       ?        ; Floating point IP
                            dw       ?        ; Floating point CS
                            dw       ?        ; Operand offset
                            dw       ?        ; Operand selector
      ST0                   dt       ?        ; ST0
      ST1                   dt       ?        ; ST1
      ST2                   dt       ?        ; ST2
      ST3                   dt       ?        ; ST3
      ST4                   dt       ?        ; ST4
      ST5                   dt       ?        ; ST5
      ST6                   dt       ?        ; ST6
      ST7                   dt       ?        ; ST7
  FPU_Struct   ENDS
      .list
    INTSEG segment at 0
      org        6*4
      INT06      dd        ?
   INTSEG ends
;------------------------------------------------------------------------
; Data segment
_DATA segment use16 para public 'DATA'
      FENV       FPU_Struct        <>
; Lista de valores de punto flotante que se usan en el test
```

Tabla 1. (continuación)

```
;---------------------------------------------------------------------------
        Op16            dt      0c06e8000000000000001h
                        dt      0c06e8000000000000010h
                        dt      0c06e8000000000000100h
                        dt      0c06e8000000000001000h
                        dt      0c06e8000000000010000h
                        dt      0c06e8000000000100000h
                        dt      0c06e8000000001000000h
                        dt      0c06e8000000010000000h
                        dt      0c06e8000000100000000h
                        dt      0c06e8000001000000000h
                        dt      0c06e8000010000000000h
                        dt      0c06e8000100000000000h
                        dt      0c06e80007ffffffffffh
    N16     equ     ($-Op16) / sizeof Op16
        Op32    dt  0c05e8000000000000001h
                        dt      0c05e8000000000000010h
                        dt      0c05e8000000000000100h
                        dt      0c05e8000000000001000h
                        dt      0c05e8000000000010000h
                        dt      0c05e8000000000100000h
                        dt      0c05e8000000001000000h
                        dt      0c05e8000000010000000h
                        dt      0c05e800000007fffffffh
    N32     equ     ($-Op32) / sizeof Op32
    D16     dw  55aah
    D32     dd  55aa55aah
        StatusWord      dw      4 dup (0)
        Results     dw  0       ; Cumulative results
        CPUIDVal dd     0       ; Results of CPUID instruction
        OrigINT06dd     0       ; Temp holding spot for INT06 vector
; Textos de salida
;---------------------------------------------------------------------------
        HeaderMsg16     db      "  16-bit Valor     FLD/FIST   FRSTOR/FIST
FLD/FISTP FRSTOR/FISTP",0dh,0ah,24h
        HeaderMsg32     db      "  32-bit Valor     FLD/FIST   FRSTOR/FIST
FLD/FISTP FRSTOR/FISTP",0dh,0ah,24h
        ValueBuf  db    "               ",24h
        Spaces          db      "  ",24h
        PassMsg   db    " BIEN ",24h
        FailMsg   db    "**FALLO*",24h
        CRLFMsg         db      0dh,0ah,24h
        PMMsg           db      "Funcionando en modo protegido 32 bits.
",0dh,0ah
                        db      "Obtendría mejores resultados en DOS
real.",0dh,0ah
                        db      "Ejecutando....",0dh,0ah,24h
        CPUIDMsg        db      "Microprocesador ID: "
        FamilyString    db      "Desconocido "
                        db      " Vendor String: "
        IDString        db      "No Detectado",0dh,0ah,24h
        Dan0411Failed   db      "*** Dan-0411 bug encontrado***",0dh,0ah,24h
        Dan0411Passed   db      "Dan-0411 no encontrado.",0dh,0ah,24h
        _DATA   ENDS
;---------------------------------------------------------------------------
```

constancia) del suceso, para después, si está permitido por el S.O., generarse una excepción (interrupción) de error del sistema operativo.

En un sistema que funcionase bien, esto permitiría que el programa detectase cuándo se produce el error y poder solucionar el problema, pero con el bug presente, el famoso bit no se pone a 1 y es imposible poder percatarse de forma automática del error.

Este fallo, según Intel, será corregido en breve en ambos modelos de microprocesador aunque, claro está, el que ya tenga uno con el "regalito" en los transistores, se va a tener que quedar el bug. Además, cómo no, se encontrará que en alguna ocasión algún programa (sobre todo si es un videojuego) sofisticado se le quedará colgado como por arte de magia en algún momento no demasiado deseable de su sesión.

La FPU puede soportar hasta 300 MIPS por segundo a 300 Mhz

Para los que sean más expertos y quieran conocer el BUG más a fondo, diremos que las instrucciones que dan problemas son concretamente FIST y FISTP cuando extraen enteros de 16 y 32 bits y que las condiciones que generan el error y los resultados del mismo son los siguientes:

1- Condiciones generadoras del error:

- Cuando se usan FIST/FISTP con enteros menores a 64 bits.
- Cuando el redondeo se programa hacia el más cercano, hacia cero o hacia arriba. En caso de tenerlo puesto como hacia el mínimo no se producirá error.
- Cuando el operando en punto flotante posee signo.

- Cuando el operando es mucho más negativo que el entero negativo que le puede equivaler.

2- Resultado obtenido en caso de error:
- Se devuelve el valor MAXNEG a la memoria, que es 8000h o 80000000h según si es dato de 16 o 32 bits.
- El bit IE (Operación inválida) no se pone a 1.
- El bit PE (Precisión error) se pone a 1.
- No se genera ninguna excepción.

3- Resultado correcto que se tendría que haber obtenido:
- Devolverse el valor 8000h o 80000000h a la memoria según longitud del entero.
- Bit IE se pone a 1.
- El Bit PE no se pone a 1.
- Generación de la excepción de error.

La polémica caché level 2

Cuando apareció el Pentium Pro ya hace bastante tiempo, lo que más destacó del mismo para todo el mundo fue que se hubiese incorporado la memoria caché level 2 integrada en el propio encapsulado del chip junto al procesador, con lo cual se podía hacer que se conectasen ambos, micro y caché L2, con un bus directo de comunicación y podía funcionar, como consecuencia de ello, a la misma velocidad que la propia CPU. El resultado era que se producía un aumento substancial del rendimiento de la caché y del sistema en general.

Pero, por lo que parece, Intel ha pensado que la tecnología que se debe incluir para hacer posible esta funcionalidad, eleva demasiado el coste del chip y ha cortado por lo sano simplemente haciendo un diseño más simple de lo explicado. El resultado ha sido el siguiente: en

La memoria caché L1 se ha duplicado respecto a su antecesor

el Pentium II, la caché también se conecta al procesador directamente, pero el bus que los comunica funciona a la mitad de velocidad que el chip. Así tenemos, por ejemplo, que si la CPU funciona a 300 Mhz, la información entra y sale de la caché a 150 Mhz, con lo que se ha creado un fantástico cuello de botella (que no es algo bueno sino malo, por desgracia, puesto que significa más lentitud).

Tecnologías paralelas

Como es fácil de entender, si se aumenta en un equipo solamente la velocidad y prestaciones de la CPU, el conjunto llega un momento que no está a la altura de la misma en cuanto a capacidad de rendimiento.

Por ello es necesario que, junto a la aparición de nuevas generaciones de microprocesadores, aparezcan mejoras en otros aspectos del hardware como son la BIOS, buses de datos, memorias, etc.

Una de las innovaciones que va a aparecer casi paralela a los nuevos Pentium II y de la cual se hablará mucho en un futuro próximo, es un nuevo modelo de BUS llamado AGP, que proviene del inglés *Accelerated Graphics Port* que podemos traducir como Puerto Acelerado de Gráficos.

Este nuevo tipo de Bus, del que sólo se podrá disponer de uno por placa base, ha sido diseñado para complementar a la tecnología actual PCI y ofrece la posibilidad de conectar la memoria RAM directamente con la placa gráfica.

El bus AGP posee varias versiones definidas, como es habitual. La primera versión, la original, fue diseñada para funcionar a 66 Mhz. La que actualmente está siendo usada, es la llamada AGPx2, que ofrece unas prestaciones de velocidad dobles a la original, conectando la placa gráfica y la memoria a 133 Mhz reales. La última versión es la llamada x4 (no hablamos de CD-ROMs) y, aunque no hay ninguna tarjeta que vaya a soportarlo de momento, ofrecerá una velocidad de transferencia de datos de 266 Mhz.

Las tecnologías paralelas como el AGP multiplicarán la potencia de los PCs

Actualmente se están presentando los primeros modelos de tarjetas gráficas, principalmente modelos con aceleración 3D, que están disponibles tanto en versión clásica PCI como en versión AGP. Un ejemplo conocido que aparecerá en el mercado español antes de fin de año es la tarjeta Matrox Millenium II.

Rendimiento del sistema

Como vamos a ver a continuación, el rendimiento de estos nuevos micros es bastante elevado en relación con los Pentium clásicos, aunque el hecho de que no haya casi modelos de generaciones anteriores a las mismas velocidades, hace que las comparaciones sean más difíciles.

La mejor referencia podemos encontrarla con el índice iCOMP(R) 2.0, comparando el Pentium MMX de 233 Mhz con el Pentium II de la misma velocidad. En este caso, la diferencia queda clara, ya que mientras el primero ofrece un valor de 203, el equivalente en

Pentium II da un resultado de 267, lo que significa una diferencia muy elevada. Casi no hace falta decir que, si la comparación la hacemos con un Pentium II a 266 Mhz, la diferencia es abismal, puesto que éste último da un resultado de 303, lo cual representaría el equivalente a un teórico Pentium MMX de unos 350 Mhz de velocidad.

Como se ha dicho antes, tampoco podemos suponer que tendremos un sistema de características generales similares, puesto que, como se ha dicho, cuando se adquiere un equipo con un procesador tan moderno es de suponer que el comprador adquiere un sistema con bus AGP, gestión de discos SCII, memoria de alta velocidad, etc., lo cual puede multiplicar varias veces el rendimiento del sistema y hacer las diferencias aún más exageradas.

Microprocesadores alternativos

Como ya se veía venir desde hace tiempo, la competencia que tienen los micros Intel cada vez es mayor. Actualmente nos encontramos que hay algún modelo de micro de la competencia que puede compararse ya con esta nueva generación y que incluye también la famosa tecnología MMX.

Ya están apareciendo micros de otras marcas de potencia casi equiparable

El último procesador de este tipo que ha aparecido es el AMD K6, que funciona a velocidades de hasta 233 y está siendo adaptado para poder ser presentados modelos de hasta 300 Mhz antes del próximo año.

Como dato final, aparte de sus buenas prestaciones, decir que su principal ventaja es que es compatible con las placas diseñadas para microprocesadores Intel normales (socket 7), lo que le da más entrada en el mercado actual frente a los Pentium II que, como hemos dicho, sólo son instalables en placas de nueva generación que dispongan del llamado SLOT ONE.

La guerra de fabricantes

Aunque parece que el mercado del microprocesador está últimamente muy tranquilo, la verdad es que los abogados de diversos fabricantes de chips están hasta arriba de papeles. La razón de ello son dos puntos diferentes:

El primer punto de enfrentamiento ha sido, por un lado, que AMD e Intel han estado mucho tiempo en los tribunales disputando si el primero tenía o no derecho a usar el término MMX en sus nuevos microprocesadores compatibles con dicha tecnología que, como sabemos, son una marca registrada de Intel. Al final, el tema ha acabado con un acuerdo en el cual AMD tendrá derecho a usar dicho término siempre y cuando lo acompañe de algo, como puede ser MMX Enhanced, MMX Compatible o similar.

El segundo punto de confrontación y que todavía no está resuelto legalmente, es la denuncia que ha puesto DIGITAL a Intel Corp. afirmando que ésta última ha copiado tecnologías patentadas por ella hace años y que las ha incorporado en sus últimos modelos de microprocesador (principalmente el Pentium Pro/Pentium II). Según Digital, las patentes tecnológicas copiadas son varias e incluyen casi todas las novedades que poseen los últimos micros Intel, como es la gestión de memoria caché y algunos aspectos de la llamada por Intel *Dynamic Execution*.

Un dato curioso

Como último dato a resaltar sobre este micro, comentaremos que Intel presentó el 28 de abril del presente año, junto a otros fabricantes, un nuevo estándar llamado *Open Arcade Arquitecture* que servirá como referencia estándar para la fabricación de máquinas recreativas y que está basada en el microprocesador Pentium II como elemento central de dicha arquitectura.

Las consecuencias de ello serán muchas, aunque quizás la más interesante será que los vídeojuegos de salón creados, podrán ser trasladados al mercado de PCs domésticos mucho más fácilmente dado que la arquitectura hard será prácticamente idéntica en ambas versiones.

Más información

Para más información sobre este procesador, puede buscarla en la propia WEB de Intel Corporation, que es la http://www.intel.com o directamente accediendo a la http://developer.intel.com, donde encontrará abundantes manuales y documentos relacionados con éste y otros micros disponibles en el llamado formato .PDF (Acrobat Reader) y en las propias páginas HTML de navegación, donde también se da bastante información.

Para poder acceder a información concreta relacionada específicamente con el Bug, se puede obtener información más exhaustiva en la propia página HTML de Intel http://developer.intel.com/design/news/flag/tech.htm y también en la dirección http://www.x86.org, donde podrá encontrar documentación sobre otros temas relacionados con funciones y áreas indocumentadas de varios micros Intel a parte de información sobre el BUG mencionado.

PROGRAMACIÓN EN VISUAL BASIC

Acceso a bases de datos con VB

Jordi Agost Moré

Los objetos de acceso a datos (DAO) posibilitan la utilización de un lenguaje de programación para acceder y manipular los datos de las bases de datos de formato .MDB y otros formatos, y administrar dichas bases de datos, sus objetos y su estructura a través del Jet Engine.

En el artículo anterior vimos cómo acceder a bases de datos mediante el control de datos. Dicho acceso nos será útil si lo que deseamos es una pequeña aplicación personal, pero en el caso de que tengamos que realizar una aplicación comercial, ésta posiblemente requerirá consultas y acciones más complejas que involucren a diversas tablas al mismo tiempo. Es aquí donde entrarán en juego los objetos de acceso a base de datos o DAO (*Data Access Objects*). Los objetos de acceso a datos nos permitirán usar un lenguaje de programación para acceder y manipular los datos de las bases de datos de formato .MDB (Access) u otros formatos incluyendo Fox, dBASE, Excel, Btrieve, Paradox y Texto delimitado (siempre que tengamos el ISAM correspondiente instalado), y administrar dichas bases de datos, sus objetos y su estructura a través del Jet Engine en sus diferentes versiones.

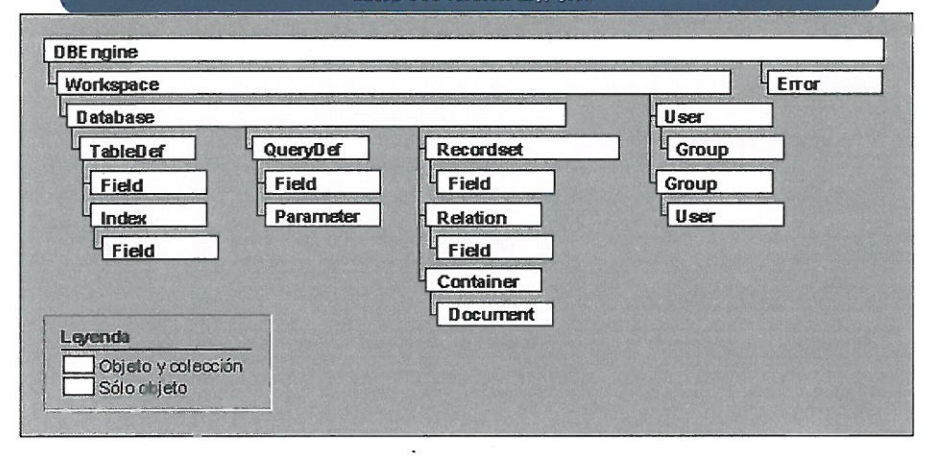
El Jet Engine

El Jet Engine es el sistema de administración de bases de datos que incorpora Access y Visual Basic y que recupera los datos y los almacena en una base de datos del sistema. A grandes rasgos diríamos que es un componente administrador de datos: los módulos que dentro de Access o Visual Basic acceden y manipulan a la base de datos. Junto con el Visual Basic 3.0 se incorporó el Jet Engine 1.0, luego apareció la actualización a la versión 2.0 del Jet Engine. Con la versión 4.0 de 16 bits Microsoft desarrolló el Jet Engine 2.5 y para la versión de VB 4.0 32 bits salió la versión 3.0 del Jet Engine. Actualmente con la versión 5.0 del Visual Basic ha aparecido la versión 3.5 del Jet Engine.

Dentro del lenguaje de Visual Basic tenemos un objeto que representa al motor Jet Engine, el DBEngine, todos los objetos de acceso a datos derivan entonces de dicho objeto. Cuando la aplicación se inicia, el motor Jet crea un objeto **DBEngine.Workspace**, dependiendo del objeto DBEngine (ver cuadro 1 de jerarquías). Dichos objetos Workspaces sirven para crear un ámbito diferente de transacción o para manejar las bases de datos con sistemas de seguridad. Si lo deseamos podemos abrir objetos Workspace adicionales cuando sea necesario, aunque por el momento no consideraremos dicho supuesto. Ya que no deseamos emplear dichos objetos, VB empleará el objeto DBEngine.Workspaces(0) de forma predeterminada. Con lo cual nos podemos olvidar de momento de ellos y trabajar directamente con bases de datos.

El siguiente objeto en importancia dentro de la jerarquía de objetos DAO es la base de datos (Database). Cualquier acceso posterior que realicemos a la base de datos se hará a partir de objetos que se habrán conectado con el objeto Database.

Cuadro 1. Ilustración del modelo de objetos de acceso a datos del motor de base de datos Jet versión 2.5/3.0.



Para abrir la base de datos utilizaremos la sentencia OpenDatabase sobre una variable que previamente habremos declarado de tipo Database (ej. Dim Db as Database). Open Database tiene la siguiente sintaxis:

```
Set base_datos =
        Workspace.OpenDatabase(nombre_bd[,
        exclusivo[,sólo-lectura[,origen]]])
```

donde:

- exclusivo es un valor de tipo Boolean que es True si la base de datos se va a abrir con acceso exclusivo (no compartido con otros usuarios) o False si se va a abrir con acceso compartido (opción por defecto).

- sólo_lectura también es un valor de tipo Boolean que será True si la base de datos se va a abrir con acceso de sólo lectura o False si se va a abrir con acceso de lectura/escritura (opción por defecto).

- origen es una expresión de cadena utilizada para abrir la base de datos. Esta cadena constituye los argumentos de conexión ODBC, que analizaremos posteriormente.

El código para abrir una base de datos sería:

```
Dim Db as Database
Set Db = OpenDatabase("c:\sp\bd.mdb",
        True, False)
MsgBox ("La base de datos abierta")
Db.Close
MsgBox "Base de datos cerrada")
```

Una vez que hemos abierto una base de datos vamos a acceder a los datos que se encuentran en ella. Para abrir una tabla de la base de datos utilizaremos el método OpenRecordset del objeto base de datos, sobre una variable que previamente hayamos definido como Recordset.

El método OpenRecordset tiene la siguiente sintaxis:

```
Set variable = base_datos.OpenRecordset(origen[,
        tipo[, opciones]])
```

donde:

- variable: es una variable que hemos definido de tipo Recordset.

- origen: será una variable de tipo String que especificará el origen de los registros del nuevo objeto Recordset. El origen puede ser un nombre de tabla, un nombre de consulta o una instrucción SQL que devuelva registros.

- tipo: especificará el tipo de recordset que queremos abrir (ver explicación posterior), sus valores posibles serán dbOpenTable para abrir un objeto Recordset de tipo tabla, dbOpenDynaset para abrir un objeto Recordset de tipo hoja de respuestas dinámica y dbOpenSnapshot para abrir un objeto Recordset de tipo instantánea.

Por ejemplo, para abrir un recordset que nos devuelva todos los registros de una tabla, escribiríamos:

```
Dim rec As Recordset, strsql As String
Set rec = Db.OpenRecordset("Libros")
If rec.RecordCount > 0 Then
  rec.MoveFirst
    While Not rec.EOF
      MsgBox rec.Fields("Titulo").Value
      rec.MoveNext
    Wend
End If
rec.Close
```

Y así obtenemos los valores que deseamos. No obstante, podemos optimizar el funcionamiento de los Recordsets eligiendo un subtipo de los recordsets. Existen tres tipos: los de tipo tabla (Table), los de tipo de hojas dinámicas (Dynaset) y los de tipo de hoja instantánea (Snapshots).

Cuando creamos un recordset sin especificar un tipo concreto (como en el ejemplo anterior), Visual Basic en principio lo intentará abrir como de tipo tabla, si no puede ser, lo intentará abrir como tipo de hoja dinámica o Dynaset, y si tampoco es posible, por último se intentará abrir como del tipo de hojas instantáneas.

Tipos de recordset

Veamos ahora más detalladamente las diferentes clases de recordset que existen.

- El recordset de tipo **tabla** (Table) es la representación en el código de una tabla concreta de la base de datos. Dicho tipo de recordset puede utilizarse para agregar, modificar o eliminar registros de una única tabla de base de datos.

Cuadro2

🕮	Sólo lectura
🕮 ✎	Lectura/escritura

Los diferentes métodos y los objetos a los que son aplicables.

Tabla	Dynaset	Snapshot	Table
AddNew	✔		✔
CancelUpdate	✔		✔
Clone	✔	✔	✔
Close	✔	✔	✔
CopyQueryDef	✔	✔	
Delete	✔		✔
Edit	✔		✔
FillCache	✔		
FindFirst	✔	✔	
FindLast	✔	✔	
FindNext	✔	✔	
FindPrevious	✔	✔	
GetRows	✔	✔	✔
Move	✔	✔	✔
MoveFirst	✔	✔	✔
MoveLast	✔	✔	✔
MoveNext	✔	✔	✔
MovePrevious	✔	✔	✔
OpenRecordset	✔	✔	✔
Requery	✔	✔	
Seek			✔
Update	✔		✔

• El tipo hoja de **respuesta dinámica** es el resultado de una consulta que puede tener registros actualizables. Un Recordset de tipo hoja de respuestas dinámica es un conjunto dinámico de registros que puede utilizarse para agregar, modificar o eliminar registros de una o más tablas de una base de datos subyacente. Este tipo de objeto Recordset puede contener campos de una o más tablas de una base de datos.

• El tipo de hojas de **respuesta instantánea** es una copia estática de un conjunto de registros que puede utilizarse para buscar datos o generar informes. Los objetos Recordset de tipo instantánea pueden contener campos de una o más tablas de una base de datos, pero no pueden actualizarse. De este modo, cuando lo que se quiera tratar sea la información de una tabla concreta, usaremos el subtipo de tabla. Si mezclamos información de diversas tablas utilizaremos los subtipos Dynaset o Snapshot, dependiendo de la consulta. Si ésta es larga y hemos de realizar diversas operaciones, utilizaremos preferiblemente Dynasets, ya que el dynaset sólo guardará el código del registro que devuelve.

En el caso de que la consulta sea corta, no debamos hacer operaciones con los registros o simplemente los tengamos que consultar muchas veces, utilizaremos Snapshots ya que éstos son una copia en memoria de todos los registros.

La gran ventaja de este tipo de Recordsets, como ya apuntábamos al principio, es que con ellas podemos hacer consultas de tipo relacional (que involucren más de una tabla), tan sólo eligiendo la sintaxis SQL apropiada (ver cuadro SQL). Aquí tenemos ejemplos de recordsets, con distintas sentencias SQL:

```
Dim rec As Recordset, strsql As String
strsql = "SELECT Libros.Titulo,
Libros.Autor,
Libros.Editorial,
Libros.Edición,
Naciones.Nacion"
strsql = strsql & "
        FROM Libros
LEFT JOIN Naciones
ON Libros.CodNacion = Naciones.CodNacion"
Set rec = Db.OpenRecordset(strsql,
        dbOpenSnapshot)
If rec.RecordCount > 0 Then
  rec.MoveFirst
    While Not rec.EOF
      MsgBox rec.Fields("Titulo").Value
& rec.Fields("Nacion")
      rec.MoveNext
    Wend
End If
rec.Close
```

Observamos que muchos de los métodos que se encuentran disponibles para el control de datos, que explicamos en el capítulo anterior, también lo están para el recordset (en el ejemplo anterior utilizamos MoveFirst, MoveNext,...). Además, los métodos disponibles variarán dependiendo del tipo de recordset que elijamos (ver cuadro 2). Además, tenemos algunos métodos nuevos e interesantes con el



SENSACIONAL OFERTA SÓLO PARA LECTORES DE SÓLO PROGRAMADORES

~~9.995~~ AHORA 3.995 ptas. c/u

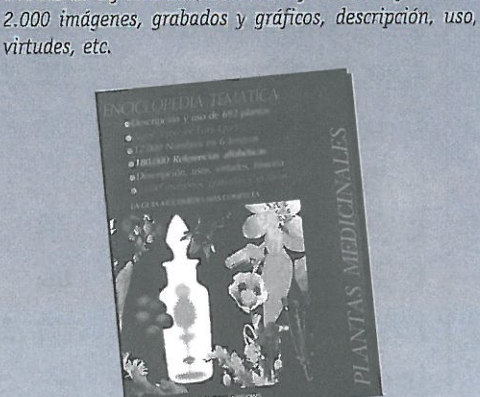


ENCICLOPEDIA DE LA PINTURA

La más completa galería de arte en CD-ROM. Permite acceder a la información por pintores, estilos, temas, períodos, nacionalidad de los pintores, ciudades y lugares donde se encuentran actualmente las obras. Incluye biografía del pintor, características de la obra, medidas, etc.

PLANTAS MEDICINALES

El Dioscórides con un nivel de interactividad absoluta, 600 Mb de información, 180.000 referencias alfabéticas, 2.000 imágenes, grabados y gráficos, descripción, uso, virtudes, etc.



PARQUES NACIONALES DE ESPAÑA

Parques Nacionales de España es una obra multimedia interactiva que facilitará a viajeros, amantes de la naturaleza, y usuarios en general, niños y mayores, el conocimiento de la fastuosa riqueza natural de España.



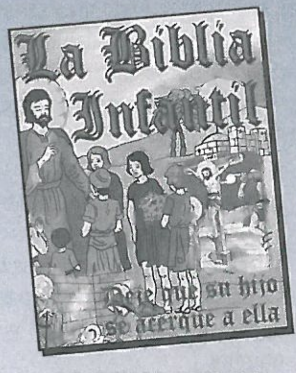
ENCICLOPEDIA MATEMÁTICA

Números, ecuaciones e inecuaciones, polinomios, geometría, sistema métrico decimal, conjuntos, divertimentos matemáticos, personajes, etc., plasmados en 800 temas expuestos en 2.800 pantallas,con más de 9.000 gráficos y 200 animaciones interactivas.

VINOS DE ESPAÑA

Más de 1.200 imágenes, 1.709 bodegas, 102 mapas autonómicos y denominación de origen, más de 1.000 plantillas de hipertexto, 23 secuencias de vídeo.





LA BIBLIA INFANTIL

Acerca a los más pequeños la religión católica, mediante ilustraciones interactivas, textos explicativos y locuciones hechas por profesionales. Una moderna herramienta de ayuda a la diversión, muy útil a la hora de hacer familia y un poco de apostolado.

VÁLIDO HASTA FIN DE EXISTENCIAS

Solicite los Programas Multimedia que desee, enviando este cupón por correo o fax (91) 661 43 86, o llamando al teléfono (91) 661 42 11*.
E-mail: suscrip@towercom.es Horario lunes a jueves de 9 a 14 y 15 a 18h. y viernes de 9 a 15h.

SOLICITO LOS SIGUIENTES PROGRAMAS MULTIMEDIA:

- ☐ Parques Nacionales de España
- ☐ Enciclopedia Matemática
- ☐ Enciclopedia de la Pintura
- ☐ Plantas Medicinales
- ☐ La Bibilia Infantil
- ☐ Enciclopedia de los Vinos de España

Nombre y apellidos .... F. Nacimiento ....
Domicilio .... C.P ....
Tel. .... Localidad ....
Provincia .... Profesión ....

FORMA DE PAGO:

- ☐ Con cargo a mi tarjeta VISA nº.................. ....
  Fecha de caducidad de mi tarjeta............... / ....
- ☐ Contra-reembolso del importe más gastos de envío y embalaje 450 ptas.
- ☐ Cheque a nombre de **CD Proyectos Especiales** que adjunto.
- ☐ Giro Postal (adjunto fotocopia del resguardo).

Firma:

TOWER COMMUNICATIONS S.R.L.
APTDO. CORREOS 54.283
28080 MADRID

objeto Recordset. Por ejemplo, el método CancelUpdate sirve para cancelar la edición en curso (hecha con el método Edit como vimos en el artículos pasado sobre el control de datos) siempre que no se haya utilizado el método Update que sirve para grabar los cambios hechos en el Recordset. El método clone nos sirve para duplicar un objeto de tipo recordset:

```
Dim rec1 As Recordset, rec2 As Recordset
Set rec1 = Db.OpenRecordset("SELECT * FROM
    Libros", dbOpenDynaset)
Set rec2 = rec1.Clone
rec1.Close
rec2.Close
```

con lo cual rec2 es ahora una copia exacta de rec1. Otro método que nos puede resultar útil es requery. Su sintaxis es:

```
Recordset.Requery
```

y sirve para que el recordset actualice los datos, volviendo a ejecutar la consulta para que así podamos tener por ejemplo, los datos más recientes. Move mueve hacia delante o hacia atrás un número determinado de registros.

Dos propiedades interesantes son:

- Index devuelve o establece el índice en un recordset de tipo tabla, por ejemplo: objeto.Index [= índicenombre]

- La propiedad devuelve o establece un valor que indica o cambia la localización aproximada del registro actual dentro del objeto Recordset, basado en un porcentaje de los registros del Recordset: objeto. PercentPosition [= valor]

Las diferentes propiedades y los objetos a los que son aplicables. Continuación.

Tabla	Dynaset	Snapshot	Table
AbsolutePosition	📖 ✎	📖 ✎	–
BOF	📖	📖	📖
Bookmark	📖 ✎	📖 ✎*	📖 ✎
Bookmarkable	📖	📖*	📖
CacheSize	📖 ✎	–	–
CacheStart	📖 ✎	–	–
DateCreated	–	–	📖
EditMode	📖	📖	📖
EOF	📖	📖	📖
Filter	📖 ✎	📖 ✎	
Index	–	–	📖 ✎
LastModified	📖	–	📖
LastUpdated	–	–	📖
LockEdits	📖 ✎	–	📖 ✎
Name	📖	📖	📖
NoMatch	📖	📖	📖
PercentPosition	📖 ✎	📖 ✎*	📖 ✎
RecordCount	📖	📖	📖
Restartable	📖	📖	False
Sort	📖 ✎	📖 ✎	–
Transactions	📖	False	📖
Type	📖	📖	📖
Updatable	📖	False	📖
ValidationRule	📖	📖	📖
ValidationText	📖	📖	📖
Value	📖 ✎	📖	📖 ✎

SQL p51

SELECT lanza consultas y permite elegir los campos que leer y su organización, pudiendo hacer diversas operaciones sobre la selección. Su sintaxis:

```
SELECT Campos
FROM tablas [WHERE Condición] [ORDER BY
    criterio]
```

Por ejemplo, para seleccionar el título y el autor de todos los libros haríamos:

```
SELECT Titulo, Autor FROM Libros
```

Si además los quisiéramos ordenados por título, haríamos:

```
SELECT Título, Autor
FROM Libros
ORDER BY Título
```

Para seleccionar todos los campos y todos los registros de la tabla libros:

```
SELECT * FROM Libros
```

También podemos poner condiciones a la hora de hacer una selección, por ejemplo si quisiéramos todos los autores de España (suponiendo que el código de España fuese 34):

```
SELECT Autor FROM Libros WHERE
CodNacion = 34
```

Otra operación que podemos realizar es, por ejemplo, contar el número de registros a través de una sentencia SQL

```
SELECT(COUNT) WHERE CodNacion = 34
```

También podemos escoger datos de diferentes tablas de la misma base de datos, con la operación JOIN. Dichos campos deben de estar unidos por una relación. Así, para seleccionar los datos del libro más la nación del autor (su descripción) haríamos:

```
SELECT Libros.Titulo, Libros.Autor,
    Libros.Editorial,
Libros.Edicion, Naciones.Nacion FROM Libros
LEFT JOIN Naciones ON Libros.CodNacion =
    Naciones.CodNacion
```

Opciones como la aplicación de filtros o la ordenación de registros, las podemos aplicar a los objetos Recordset (de tipo Dynaset o Snapshot). Dichas opciones las introduciremos como cadenas, que serán las restricciones que queramos. Por ejemplo, la propiedad filter nos permitirá definir cadenas tal que expresen las condiciones que deben cumplir los registros a mostrar:

```
Set ds = Db.OpenRecordset("Libros",
    dbOpenDynaset)
Filtro$ = "CodNacion > '20'
    AND CodNacion < '40' "
ds.Filter = Filtro$
```

La propiedad Sort sirve para establecer un criterio de ordenación por uno o más campos. El criterio se establecerá igual que en la propiedad Filter, es decir, con una cadena alfanumérica que especificará el orden de los registros que mostrar.

Ejecución sentencias SQL

En SQL existen una serie de sentencias que podemos ejecutar directamente desde VB. Dichas sentencias son:

INSERT

Nos servirá para insertar un nuevo registro en la base de datos:

```
INSERT INTO Libros (Titulo,Autor)
VALUES ('Las cosas de Casa', 'Pérez Fernández')
```

UPDATE

Modificará los registros de la base de datos. Por ejemplo, para poner todos los códigos de nación de libros a 00:

```
UPDATE Libros SET CodNacion = 00
```

DELETE

Esta instrucción borra registros de una base de datos, por ejemplo:

```
DELETE FROM Libros WHERE CodNacion = 34
```

Estas sentencias SQL las podemos procesar directamente con la orden Execute. Su sintaxis es la siguiente:

```
base_datos.Execute origen[, opciones]
```

• origen: cadena de tipo SQL que incluirá las sentencias a ejecutar.

• opciones puede tener como valor las siguientes constantes:
- DbDenyWrite. Deniega el permiso de escritura al resto de usuarios.
- DbInconsistent (por defecto). Permite actualizaciones inconsistentes.
- DbConsistent. Obliga a que las actualizaciones sean consistentes (excluyente del anterior).
- DbSQLPassThrough. Paso a través de SQL. Para ODBC.
- DbFailOnError. Deshará las actualizaciones en caso de error.
- DbSeeChanges. Generará un error en tiempo de ejecución si otro usuario modifica los datos en edición.

Debemos tener en cuenta que el método Execute sólo es válido para las consultas de acciones. Si utilizamos Execute con otro tipo de consultas, se producirá un error. Debido a que las consultas de acciones no devuelven registros, Execute no devuelve un conjunto de registros. Por ejemplo:

```
Db.Execute "INSERT INTO Libros (Titulo,Autor)
 VALUES ('Las cosas de Casa', 'Perez Fernández') "
```

Consultas

En una BD de Access podemos almacenar tablas, consultas y formularios. Éstos no los podremos visualizar desde VB pero sí podremos visualizar las consultas, que almacenan los datos que utilizamos a menudo y en Access se llaman QueryDef. Si queremos insertar una nueva consulta en una base de datos de Access y desde VB, haríamos (si el objeto base de datos está abierto y se llama Db):

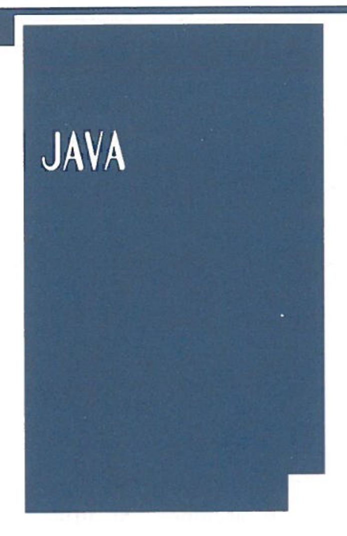
```
Dim Cons As QueryDef, strsql As String
strsql = "SELECT * FROM Libros"
Set Cons = Db.CreateQueryDef("Consulta de
    prueba", strsql)
Cons.Close
```

Para recuperar los datos podemos realizar un recordset sobre una consulta. Para borrar la consulta que hemos creado:

```
Db.DeleteQueryDef ("Consulta de prueba")
```

También podremos hacer una copia del objeto:

```
Db.CopyQueryDef
```



Packages, clases anidadas y algún grano más

Óscar Prieto Blanco

El aprendizaje del hombre suele pasar por tres etapas.
En la primera aprende las respuestas correctas.
En la segunda, a hacer preguntas.
En la tercera y última, qué preguntas vale la pena plantear.

Bits and Pieces.
Revista estadounidense.

Java, como herramienta ligada cada vez más inextricablemente al desarrollo de las comunicaciones vía red (ya sea Internet, *intranets*, etc.), está sirviendo como catalítico de un montón de tecnologías. El fin último de todas ellas es, simplemente, un intento de dominar cada vez mejor el inmenso caudal de información a que tenemos acceso y que nos bombardea incesablemente. Aún así, es todavía tarea nuestra el aprehender dicho conocimiento y el efectuar las preguntas oportunas.

Un poeta cubano dijo una vez: "Un grano de poesía es suficiente para endulzar un siglo". En el artículo de este mes, nos vamos a dedicar a conocer ciertos granos de Java, que se nos han ido cayendo del saco en el discurrir de la serie y que, todos juntos, todavía son capaces de suministrarnos una exquisita taza de café.

Packages y especificadores de acceso

A lo largo de estos artículos he venido utilizando los especificadores de acceso un poco alegremente, así que ya va siendo hora de ser algo más formal. Para ello debemos comprender el concepto de *package*.

En el transcurso de las distintas fases de desarrollo de un programa, más tarde o más temprano nos encontramos con el problema de organizar el incipiente número de clases que vamos codificando.

En Java podemos agrupar un conjunto de clases relacionadas mediante el uso de los *packages*. El término *package* es el equivalente a la idea de librería de componentes reutilizables, existente en otros lenguajes.

En Java agrupamos un conjunto de clases relacionadas mediante la utilización de los packages

Estudiemos un poco de código con objeto de introducirnos sin traumas en este tema.

```
/////////////////////////////
// fichero Shark.java
/////////////////////////////
// Debe ser, invariablemente, la
// primera línea del fichero.

package PeligrosVerano;

public class Shark
```

```
{
  int numDient;
  private String nombre;
  protected String color
  public Shark(int d, String n, String c)
  {
      numDient=d;
      nombre=n;
      color = c;
      P.prt("Tengo hambre");
  }
  //...etc
}
// Fin de fichero Shark.java

/////////////////////////////
// fichero Octopus.java
/////////////////////////////
package PeligrosVerano;

public class Octopus
{
  int numTenc;
  private String name
  private protected int longitud;
  public Octopus(int n, String s, int l)
  {
      numTenc=n;
      name=s;
      longitud=l;
      P.prt("Soy un pulpo de la pradera");
  }
  //... etc
}

class P
{
  static prt(String s)
  {
      System.out.println(s);
  }
}
//Fin de fichero Octopus.java
```

En el ejemplo anterior hemos definido el *package* PeligrosVerano, para lo cual han sido utilizados dos ficheros en los que aparecen tres clases definidas: Shark, Octopus y P. A la hora de crear un *package*, debemos tener presentes las siguientes ideas:

Las librerías en Java se denominan packages

• La palabra clave *package* debe ser la primera sentencia que aparezca en el fichero (exceptuando, claro está, los espacios en blanco y comentarios).

• Es aconsejable que todas las clases que vayan a ser incluidas en un *package* se encuentren en el mismo directorio. Esta recomendación nos la podemos saltar a la torera, pero corremos el riesgo de encontrarnos con determinados problemas difíciles de resolver a la hora de compilar, en el supuesto caso de que no hilemos muy fino.

• Ante todo, recordar que en un fichero únicamente puede existir, como máximo, una clase con el especificador de acceso *public*, debiendo coincidir, en este caso, el nombre del fichero con el nombre de la clase (respetando, eso sí, las capitalizaciones).

Si ahora decidiera integrar este *package* en otro programa (imaginemos que el directorio PeligrosVerano se encuentra dentro de C:\Utiles), la primera tarea a ejecutar sería añadir la ruta de acceso a la variable de entorno CLASSPATH:

```
CLASSPATH= .;
d:\JAVA\LIB;
C:\UTILES
```

A partir de este momento, en un programa que estuviera desarrollando podría acceder a las clases que contiene PeligrosVerano utilizando la sentencia *import*, por ejemplo:

```
// Fichero Oceano.java
import PeligrosVerano.Shark ;

public class Oceano
{
  Shark Donatello;
  // Si no utilizo la sentencia import.
  // Esta notación es la única opción,
  // cuando utilizo varios packages en
  // los cuales existen definidas clases
  // con el mismo nombre, que podrían
  // conformar un código ambiguo.
  PeligrosVerano.Octopus Rubens;
  //... etc
}
```

En el ejemplo de esta sección han sido incluidos todo tipo de especificadores de acceso, así que estudiémoslos detenidamente.

Modificadores de acceso a clases

Existen dos:

"friendly" es el que tenemos por defecto si no especificamos nada en la definición de la clase.

Significa que esta clase es accesible a todas las demás clases contenidas en el *package*, pero no puede ser accedida de ninguna forma por el usuario de dicho *package*. Las clases que contienen esta especificación normalmente tienen como misión ayudar a las clases públicas a efectuar sus cometidos, es decir, son útiles para el programador del *package* pero no es deseable o conveniente su manipulación por parte de los clientes del *package*.

En el ejemplo visto, las clases *Shark* y *Octopus* pueden acceder a la clase P pero los usuarios del *package* (*clients programmers*) no.

"public", todas las clases definidas en el *package* bendecidas con el especificador de acceso *public* podrán ser accedidas por el usuario de dicho *package*.

Con la aparición de las clases anidadas han sido añadidos más especificadores de clase, que serán tratados posteriormente en este artículo.

Modificadores de acceso a miembros de clases

"friendly", con el que contamos por defecto si no especificamos nada en las variables o en los métodos de la clase. Los miembros con este tipo de acceso pueden ser accedidos:

1. Por todas las clases que se encuentren en el fichero donde se encuentre también definida la clase.

2. Si nos encontramos en un *package* por todas las demás clases que lo integren.

3. Y por último, si la clase se encuentra en un fichero que no forma parte de ningún package explícitamente indicado, las clases pertenecientes a otros ficheros que se encuentren en el mismo directorio también tendrán acceso a dichas variables. (En java, todos los ficheros de un directorio que no tengan explícitamente indicado el nombre de un *package* son reunidos en un *package* sin nombre, es decir, entre las clases en ellos contenidas existe acceso friendly).

Por Ejemplo:

```
// Acuario.java
class Acuario
{
public static void main(String args[])
    {
        // El compilador no da error:
        Piraña x = new Piraña();
        x.f();
    }
}

// En otro fichero en el mismo directorio:
// CU2.java
class Piraña
{
    void f()
    {
        System.out.println("¡Ñan, ñan!");
    }
}
```

"public", estos son los únicos miembros que tienen permiso para ser accedidos por los usuarios de la clase. Se suele decir que los miembros públicos conforman la interfaz de la clase con el usuario. De esta forma, en el ejemplo visto al comienzo de esta sección, al único miembro que podemos acceder los usuarios del *package* Peligros Verano, en relación a la clase *Shark*, es su constructor.

"private", nadie puede acceder a estos miembros, excepto la clase que los contiene dentro de sus métodos. Conseguimos máxima restricción en el acceso, pero a diferencia del C++, en Java es poco utilizada esta variante, ya que con el modificador por defecto *friendly*, conseguimos un adecuado grado de encapsulación.

"protected", cuando queremos refinar una clase perteneciente a un *package* aplicando el mecanismo de la herencia, desde fuera de dicho *package*, a los únicos miembros a los que tendremos acceso serán a los públicos de la clase elegida como base. A veces, al creador de la clase base le gustaría tomar algún miembro en particular y permitir su acceso a las clases derivadas, pero no al mundo en general.

Para resolver esta papeleta contamos con la *keyword protected*. Así si tomamos como clase base la clase *Shark*, tendríamos acceso en la derivada tanto a su constructor como al atributo *String* color. Un detalle que no podemos y no debemos olvidar en ningún momento se refiere a la circunstancia de que dentro de un *package* un miembro de clase *protected* se comporta de manera similar, en cuanto a derechos de acceso, que un miembro *friendly*.

"private protected", será utilizado en el caso de que deseemos que un miembro de una clase pueda ser accedido por sus clases derivadas y por nadie más (negamos el acceso, incluso, a las clases pertenecientes a su *package*).

Gracias a la utilización de esta combinación obtenemos un enorme control sobre la disponibilidad de un elemento para el resto del mundo, tal vez por esa misma razón es raramente empleada.

Clases contenedoras e iteradores

Los *arrays* conforman la estructura más eficiente con que cuenta Java para almacenar y acceder a objetos (es decir, *handles*). Entre sus ventajas se pueden citar: el hecho de almacenar objetos de un tipo conocido y también que podemos guardar en ellos los tipos básicos (que no son objetos). Como principal desventaja nos encontramos con que deben tener un tamaño fijo (ya sea fijado en tiempo de compilación o en tiempo de ejecución).

Los arrays deben tener un tamaño fijado en tiempo de compilación o en tiempo de ejecución

En la mayoría de las ocasiones, deberemos trabajar con un conjunto de objetos cuyo número exacto sólo será conocido en tiempo de ejecución (*runtime*). Par

resolver esta eventualidad Java cuenta con las siguientes clases contenedoras: *Vector, BitSet, Stack* y *Hashtable*. Estas poquitas clases incluyen entre sus capacidades la propiedad de expandirse automáticamente para poder acomodarse al arbitrario número de objetos que vayan a ser contenidos.

Pero, como en la vida, no todo son ventajas y hay que leer la letra pequeña que aquí aparece en forma de ciertas restricciones, muy a tener en cuenta si no queremos excepciones revoloteando sobre nuestro código.

Dado que en Java no existe ninguna construcción en el lenguaje que soporte un tipo de dato genérico (C++ soporta tipos parametrizados mediante la palabra reservada *template*), todas estas clases guardan sólo objetos tipo *Object* (por tanto, no podemos utilizar las clases contenedoras con los tipos básicos del lenguaje, deberemos utilizar sus clases equivalentes).

Siempre podemos asignar un objeto de una clase derivada a un objeto de una de sus clases base

En este punto hay que recordar que todas las clases en Java derivan en última instancia de la clase *Object*, lo que implica que a un *handle* de tipo *Object* le podemos asignar un *handle* de cualquier clase que hallamos declarado. Esta operación se conoce como *upcasting* y significa que siempre podemos asignar un objeto de una clase derivada a un objeto de una de sus clases base. El problema que puede aparecer está relacionado con la pérdida de información cuando guardamos un objeto en un *container*.

Veámoslo con un ejemplo muy simple de entender:

```
Date d = new Date();
// casting implícito
Object obj = d;
//...
// Obtenemos un error obj
// es un handle a Object:
System.out.println("Día semana:"
                        + obj.getDay());
// La forma correcta
// (casting explícito):
System.out.println("Día semana:" +
// obj visto como un handle a Date
  ((Date)obj).getDay());
```

Como podemos observar en el anterior fragmento, siempre hay que recordar la aplicación del operador de *casting* si no queremos perder información (a esta operación realizada se la conoce con el término *downcasting*).

Retomemos el tema de los *containers* y veamos el uso de la clase *Vector*. Un *Vector* es similar a un *array* pero con la diferencia de que puede incrementar su tamaño dinámicamente.

Por ejemplo:

```
// Fechas.java
import java.util.*;

public class Fechas
{
// clase anidada

class Printer
{
   void printAll(Enumeration e)
   {
      while(e.hasMoreElements())
        System.out.println(
      e.nextElement().toString());
   }
  }

 public static void main(String args[])
 {
  Vector v = new Vector();

  // Añadimos unos elementos:
  v.addElement(new Date());
  long l = System.currentTimeMillis();
  v.addElement(new Date(l));
  l =- 5000000;
  v.addElement(new Date(l));

  // Utilización de iteradores
  Fechas f = new Fechas();
  Fechas.Printer P = f.new Printer();
  P.printAll(v.elements());
  }
}
```

En el listado anterior nos tropezamos con el concepto de iterador (llamados *enumerators* en Java). Su cometido consiste en proveer de un grado más de abstracción al manejo de los *containers*. Gracias a esta clase podemos movernos a través de una secuencia de objetos y seleccionar cada objeto en esa secuencia sin necesidad de conocer o de preocuparnos de su estructura interna (esta facultad añade gran flexibilidad al proceso de diseño de nuestras clases). Esto es, nos importa bastante poco el hecho de que utilicemos un *Vector*, un *Stack*, una *Hastable*, u otra estructura, para contener nuestros objetos.

La técnica de trabajo con iteradores se reduce a obtener un objeto de tipo *Enumeration* mediante la llamada a un método llamado *elements()*, que es soportado por todas las clases contenedoras. Acto seguido, la llamada a *nextElement()* nos retorna el siguiente objeto en la secuencia. Adicionalmente contamos con el método *hasMoreElements()* que nos avisa cuando ya hemos atravesado toda la secuencia.

Como introducción a otras clases contenedoras, en el listado1 he incluido la implementación de una lista enlazada y en el listado2 la implementación de una clase que simula a un conjunto (ambos fuentes se incluyen en el CD-ROM como List.java y Set.java respectivamente).

Como nos encontramos en verano, os incluyo el siguiente pasatiempo, que está relacionado con la utilización de los *stacks*: escribir un programa en el cual una sola variable, de tipo char, esté definida. El programa tiene que leer una línea de texto (cuya longitud es arbitraria) y escribir dicha línea al revés. Por ejemplo, si

la entrada fuera ABC, la salida deseada sería CBA.

La solución se encuentra al final del artículo, pero espero que os rompáis un poco la cabeza antes de ojearla.

En la clase Fechas, nos encontramos con la definición de una clase anidada (*inner class*), cuya notación fue introducida en el artículo anterior, para poder aplicar el nuevo modelo de manejo de eventos añadido a la versión 1.1 del JDK. Así que ya va siendo hora de estudiar más detenidamente este aspecto del lenguaje.

Clases anidadas (*inner classes*)

A partir de la versión 1.1, en Java las clases pueden ser definidas en cualquier alcance. Esto incluye la definición de clases como miembros de otras clases; definición de clases dentro de un método; definición dentro de un bloque de sentencias; e, incluso, definición de clases dentro de una expresión (clases anónimas). Veamos un primer ejemplo:

```
Class computer
{
  // La memoria de la computadora
  private byte mem[] = new byte[256];
  // Los registros del micro
  class microReg
  {
      byte reg[] = new byte[4];
      byte IP;
      public microReg(byte instpointer)
      {
        IP = instpointer;
      }
      public void seIP(byte ipnew)
      {
        IP = ipnew;
        public void setReg(int index, byte cont)
        {
          reg[index&0x3] = cont;
        }
        public void storeMem(int regIndex,
                    int dirMem)
```

Listado 1

```
//List.java: implementación de una lista enlazada
//Una clase para los nodos de la lista
class List Node{
Object data; //Dato a guardar
ListNode next; //Puntero al siguiente elemento
//Constructor : para el nodo final de la lista
ListNode(Object obj){
        data = obj;
        next = null; // No hay ninguno próximo}
// Constructor: Para nodos intermedios o iniciales
ListNode(Object obj, ListNode nextNode){
        data = obj, next = Node;}
        //Retornamos el obj de este nodo
        Object getObject (){return data;}
}
//Retornamos el siguiente nodo
        ListNode getnext(){return next;}
} // Fin de la clase ListNode
//Definición de la clase lista
public class List{
    private ListNode firstNode;
    private ListNode lastNode;
    private String name: //nombre de la lista
  // Construimos una lista vacía
  public List( String s ){
     name = s; firstNode = lastNode = null;}
  // Otro constructor
  public List() { this( "lista" ); }
  // Insertamos un objeto en la cabeza de la lista.
  // Si la lista está vacía, firstNode y lastNode se refieren al mismo objeto. De otra manera, a
  // firstNode le asignamos el nuevo nodo.
  public void insertAtFront( Object insertItem ){
     if ( isEmpty() ) firstNode = lastNode = new ListNode( insertItem );
     else  firstNode = new ListNode( insertItem, firstNode );}
  // Insertamos un objeto en la cola de la lista.
  // Si la lista está vacía, firsNode y lastNode se  refieren al mismo objeto. De otra manera,
  // el handle next de lastNode apuntará al objeto introducido.
  public void insertAtBack( Object insertItem ){
     if ( isEmpty() )
        firstNode = lastNode = new ListNode( insertItem );
     else  lastNode = lastNode.next = new ListNode( insertItem );}
  // Eliminamos el primer nodo de la lista.
  public Object removeFromFront() throws EmptyListException{
     Object removeItem = null;
```

(Continuación Listado 1)

```
    // Si está vacía arrojamos una excepción
    if ( isEmpty() )  throw new EmptyListException( name );
    removeItem = firstNode.data;
    // Si sólo existe un elemento
    if ( firstNode.equals( lastNode ) ) firstNode = lastNode = null;
    else  firstNode = firstNode.next;
    return removeItem;
  }
  // Eliminamos el último nodo de la lista
  public Object removeFromBack() throws EmptyListException
  { Object removeItem = null;
    // Si está vacía arrojamos un execpción
    if ( isEmpty() )
      throw new EmptyListException( name );
    removeItem = lastNode.data;
    // Si sólo existe un nodo
    if ( firstNode.equals( lastNode ) )
      firstNode = lastNode = null;
    else {
      ListNode current = firstNode;
      while ( current.next != lastNode )
        current = current.next;
      lastNode = current;
      current.next = null;}
    return removeItem;}
  // ¿Está vacía la lista?
  public boolean isEmpty() { return firstNode == null; }
  // Impresión de los contenidos de la lista
  public void print(){
    if ( isEmpty() ){
      System.out.println( name+" vacía ");
      return;}
    System.out.print("La " + name + " es: ");
    ListNode current = firstNode;
    while ( current != null ) {
      System.out.print( current.data.toString() + " " );
      current = current.next;}
    System.out.println();  System.out.println();}
}
// Una excepción casera
class EmptyListException extends RuntimeException {
  public EmptyListException( String name )
  { super( "La " + name + " está vacía" );  }
}
```

```
      {
        mem[dirMem&0xFF] =
                  reg[regIndex&0x3];}
        public void loadMem(int regIndex,
                        int dirMem)
      {
        reg[index&0x3] = mem[dirMem&0xFF];
      }
  }

  // Para simplificar la notación
  public microReg regs(byte dirInst)
  {
    return new microReg(dirInst);
  }

  public static void main(String args[])
  {
      computer comp = new computer();

      // Para utilizar instancias de clases
      //anidadas, necesito instancias de sus
      //clases contenedoras.
      computer.microReg reg =
          comp.new microReg(0x00);
      // Más intuitivamente:
      computer.microReg reg_bis =
          comp.regs(0x00);
  }
}
```

Lo primero que debemos comprender es la diferencia entre las clases anidadas y la composición de clases. Si queremos crear un objeto de una *inner class*, debemos codificarlo expresamente utilizando la fea y oscura notación mostrada en el ejemplo anterior.

A partir de la versión 1.1, las clases pueden ser definidas en cualquier alcance

Normalmente, la clase contenedora (*outer class*) poseerá un método que retorne un *handle* a una *inner class*, como podemos observar con la función *regs()* del ejemplo, lo cual simplifica un poco la notación.

¿Qué ventajas nos ofrecen las clases anidadas, para que resulte de alguna utilidad tener que preocuparnos con estas notaciones tan extrañas? Pues las ventajas de su utilización son las que explicamos a continuación:

- Una *clase anidada* puede acceder a los miembros de clase o de instancia de su clase contenedora o, en el caso que se encuentre definida dentro de un bloque,puede acceder a las variables locales definidas dentro de dicho bloque.

Una clase anidada puede acceder a los miembros de clase o de instancia de su clase contenedora

- Pueden ser utilizadas para conseguir un sofisticado mecanismo de encapsulación. Esta característica se puede advertir en la siguiente variación del código de ejemplo que proporcionamos al comienzo de esta sección:

```
package prueba;

interface Registers
{
  void setIP(byte ipNew);
  void setReg(int nInd, byte contents);
  void storeMem(int regInd, int dirMem);
  void loadMem(int regInd, int dirMem);
}

class computer
{
  // La memoria de la computadora
  private byte mem[] = new byte[256];
  // Los registros del micro
  // Ojo al calificador private
  private class microReg implements Registers
  {
    byte reg[] = new byte[4];
    byte IP;
```

Listado 2

```
// Set.java  Clase que encapsula el concepto de conjunto

public class Set extends Object {
  // Los elementos del conjunto
  protected Object elements[];
  // Constructores
    protected Set() { }
  public Set(Object array[]) {
    elements=new Object[array.length];
      for(int i=0;i<elements.length;i++)
        elements[i]=array[i];
  }
  // El numero de elementos del conjunto
    public final int size()
    { return elements.length; }
  // Obtención de un elemento del conjunto
  public final Object getElement(int n)
    { return elements[n]; }
  // La unión de dos conjuntos
  public Set union(Set s) {
    boolean unique;
    int k;
    Object uObj[],
    tempObj[]=new Object[elements.length+s.size()];
    if(elements.length > s.size()) {
      for(k=0;k<s.size();k++)
        tempObj[k]=s.getElement(k);
      for(int i=0;i<elements.length;i++) {
        unique=true;
        for(int j=0;j<s.size();j++) {
          if(elements[i].equals(tempObj[j]))
              unique=false;
        }
        if(unique)
          tempObj[k++]=elements[i];
      }
    } else
      {
      for(k=0;k<elements.length;k++)
        tempObj[k]=elements[k];
      for(int i=0;i<s.size();i++) {
        unique=true;
        for(int j=0;j<elements.length;j++){
          if(s.getElement(i).equals(tempObj[j]))
              unique=false;
```

(Continuación Listado 2)

```
            }
            if(unique)
                tempObj[k++]=s.getElement(i);
        }
    }
    uObj=new Object[k];
    for(k=0;k<uObj.length;k++) {
        uObj[k]=tempObj[k]; .
    }
    return new Set(uObj);
}
// Intersección de dos conjuntos
public Set intersection(Set s) {
    boolean unique; int k=0;
    Object tempObj[],iObj[];
    if(elements.length > s.size()){
        tempObj=new Object[elements.length];
        for(int i=0;i<elements.length;i++){
            unique=true;
            for(int j=0;j<s.size() && unique;j++){
                if(elements[i].equals(s.getElement(j))){
                    tempObj[k++]=elements[i];
                    unique=false;
                }
            }
        }
    }
    else {
        tempObj=new Object[s.size()];
        for(int i=0;i<s.size();i++) {
            unique=true;
            for(int j=0;j<elements.length && unique;j++){
                if(s.getElement(i).equals(elements[j])){
                    tempObj[k++]=elements[j];
                    unique=false;
                }
            }
        }
    }
    iObj=new Object[k];
    for(k=0;k<iObj.length;k++)
        { iObj[k]=tempObj[k]; }
    return new Set(iObj);
} // Fin función intersección
}// Fin clase Set
```

```
    // constructor privado
    private microReg(byte instpointer)
    {
      IP = instpointer;
    }
    public void seIP(byte ipnew)
    {
      IP = ipnew;
      public void setReg(int index, byte cont)
      {
        reg[index&0x3] = cont;
      }
      public void storeMem(int regIndex,
            int dirMem)
      {
      mem[dirMem&0xFF] =
                    reg[regIndex&0x3];
    }
    public void loadMem(int regIndex, int dirMem)
    {
      reg[index&0x3] = mem[dirMem&0xFF];
    }
  }
  // En este caso es necesaria
  public microReg regs(byte dirInst)
  {
    return new microReg(dirInst);
  }

  public static void main(String args[])
  {
      computer comp = new computer();
      Registers r = comp.regs(0x00);
  }
}
```

En el anterior fragmento de código, la clase anidada es “PRIVADA” (hay que puntualizar que sólo clases anidadas pueden ser calificadas como *private* o *protected).*

Como principal implicación de este hecho, nos encontramos con que un usuario de este *package* no puede crear un objeto *microReg* es decir, tiene en todo caso restringido el acceso y el conocimiento a dicho miembro. Únicamente dispondría de la posibilidad de acceder a los miembros declarados en la interfaz que dicha clase implementa. De esta manera al diseñador de la clase anidada se le facilita la acción de esconder los detalles de la implementación de dicha clase.

Clases anidadas estáticas

Si nuestra clase anidada no necesita acceder a ninguna de las variables de instancia de su clase contenedora, la necesidad de crear un objeto de dicha clase para poder acceder a nuestra clase anidada, parece algo cuando menos poco inteligente. En este caso concreto, contamos con la posibilidad de especificar a nuestra clase como *static*, no necesitando en esta ocasión, ningún objeto de la clase contenedora para poder instanciar un objeto de la clase anidada.

Un ejemplo que clarifique lo anterior podría ser:

```
//VampireTheatre.java
interface Vampire
{
 void print();
}

class VampireTheatre
{
  // Siempre fue muy independiente
  static class Lestat implements Vampire
  {
    public void print()
    {
      System.out.println(“Temblad malditos”);
    }
  }

  // Su amigo un poco más tímido
  private static class Louis implements Vampire
  {
    public void print()
    {
      System.out.println(“Os amo, mortales”);
    }
  }
  static public Louis getVampire()
  {
    return new Louis();
  }

  // Un vampiro más clasico
  class Armand implements Vampire
  {
    public void print()
    {
      System.out.println(“La tradición ...”);
    }
  }

  // La función principal:
  public static void main(String args[])
  {
    Vampire Lest = new Lestat();
    Vampire Loui =
          VampireTheatre.getVampire();
    VampireTheatre vt = new VampireTheatre();
    Vampire Armd = vt.new Armand();
    Lest.print();
    Loui.print();
    Armd.print();
  }
}
```

Clases anidadas en funciones y alcances

Todavía no hemos acabado con las sutilezas sintácticas del anidamiento de clases. En este apartado queremos aprender que también podemos definir clases dentro del ámbito de una función, dentro de un bloque de sentencias y, como colofón, el atajo (*shorthand*), conocido como clases anónimas, con el que definimos una clase dentro de una expresión.

```
//VampireTheatrebis.java

interface Vampire
{
 void print();
}

class VampireTheatrebis
{
 // Anidamiento dentro de una función:
 public Vampire getLouis()
 {
    class Louis implements Vampire
    {
      public void print()
      {
        System.out.println(“Soy Louis”);
      }
    }
    return new Louis();
 }

 // El atajo conocido como clase anónima
 public Vampire getLestat()
 {
   return new Vampire()
   {
     public void print()
     {
       System.out.println(“Soy Lestat”);
     }
   };
   // OJO al punto y coma. Marca el fin de
 // la expresión que contiene a la clase
 // anónima (en este caso el return)
 }

 // Anidamiento dentro de un alcance:
 public void newMonster()
 {
   int test;
   if((test = (int)(Math.random()*2)) == 0)
   {
     class Frankie
     {
       private int age;
       Frankie(int ag)
       {
         age=ag;
       }
       public int getAge()
       {
         return age;
       }
     }
     System.out.println(“Edad de Frankie: “+
        new Frankie(220).getAge() + años);
   }
 }

 // La función principal:
 public static void main(String args[])
 {
     VampireTheatrebis vtbis =
     new VampireTheatrebis();
     Vampire Lest = vtbis.getLestat();
     Lest.print();
     Vampire Loui = vtbis.getLouis();
     Loui.print();
     vtbis.newMonster();
 }
}
```

En la función *getLestat()* nos encontramos la definición de una clase anónima. En ella podemos ver cómo una expresión *new* puede finalizar con el cuerpo de una clase. Cuando el compilador se encuentra

Una clase anónima puede contener inicializadores pero no puede poseer un constructor

con dicha sintaxis, toma la clase (o *interface)* posterior al *token new* y la deriva (o implementa) con el cuerpo especificado. De manera automática el resultado de esta expresión *new* sufre un *upcasting,* a la clase o *interface* tomados como base.

Con esta técnica obtenemos un efecto similar al conseguido definiendo una clase anidada con un nombre dentro de ese mismo alcance. Esta es la única situación en la que el nombre de una *interface* puede seguir legalmente a la *keyword new.* En estos casos la lista de argumentos deberá ser siempre nula, para coincidir con el constructor de la superclase de la *interface*, que siempre será *Object.* De una manera más general una clase anónima puede contener inicializadores pero no puede poseer un constructor. La lista de argumentos, asociada con la expresión *new*, será implícitamente pasada al constructor de la superclase.

Relación entre clases anidadas y contenedoras

Después de revisar toda esta parafernalia y pirotecnia de trucos, sintaxis un poco extrañas, etc, vistas en relación con las clases anidadas, ¿qué es lo que tendremos que recordar para poder trabajar con ellas adecuadamente? La única idea que es necesario tener bien clara es que una clase anidada puede acceder a cualquiera de los miembros de su clase contenedora (en el lenguaje C++ no sucede de esta forma).

Una clase anidada debe mantener algún tipo de handle al objeto particular de su clase contenedora responsable de crearla

De esta afirmación podemos deducir que una clase anidada debe mantener algún tipo de *handle* al objeto particular de su clase contenedora que ha sido el responsable de crearla. Por suerte el compilador se ocupa de todos los detalles. Aún así, esto debiera servir para aclarar por qué una clase anidada necesita una instancia de su clase contenedora y por qué una clase anidada estática no la necesita, así como no puede, por lo tanto, acceder a los miembros de su clase contenedora.

Conclusión

En este artículo me he dedicado a asentar algunos aspectos del lenguaje que no habían sido tratados todavía con la profundidad necesaria en esta serie. De esta manera hemos dado un repaso integral a los *packages* y a su influencia en los modificadores de acceso, cuyo significado exacto ha quedado suficientemente fijado en este momento. Posteriormente nos introducimos en el mundo de las clases *containers* de Java, lo que nos ha servido para entender el potente y, a la vez, simple concepto de iterador.

Por último, hemos dado un buen repaso a la problemática de las clases anidadas, cuyo entendimiento es fundamental para poder utilizar el nuevo modelo a eventos incluido en la versión 1.1 del JDK.

Solución al pasatiempo

Me imagino que os habréis divertido resolviéndolo. Aquí os enseño una posible solución, ¿es posible que exista alguna más?

```
// Reverse.java
import java.io.IOException;
public class Reverse
{
  static void readWrite() throws IOException
  {
        char ch = (char)System.in.read();
        if(ch != '\n')
        {
          readWrite();
          System.out.print(ch);
        }
  }

  public static void main(String args[])
            throws IOException
  {
         System.out.println("Introduce una línea:");
         Reverse.readWrite();
         System.out.println();
  }
}
```

Contactar con el autor

Si se desea formular cualquier duda, comentario, sugerencia o crítica sobre el contenido de los artículos, se anima al lector a ponerse en contacto en la dirección:

E-Mail myoprieto@redestb.es

Protección de datos

Enrique Díaz Trobo

"Si algo puede ir mal, irá mal". Ningún administrador de red puede ignorar esta máxima. En este artículo relataremos algunas técnicas para hacer frente a las leyes de Murphy, a la mala pata o a la mala idea.

Se dice que en programación sólo se hacen los programas como Dios manda cuando de ellos depende el dinero o vidas humanas. Y por este orden. Pues bien, en la protección de nuestros datos va a pasar lo mismo. Cuanto más valiosos sean los datos que tenemos en la red mejor protegidos estarán, ya que la protección de los datos puede ser de vital importancia para la empresa.

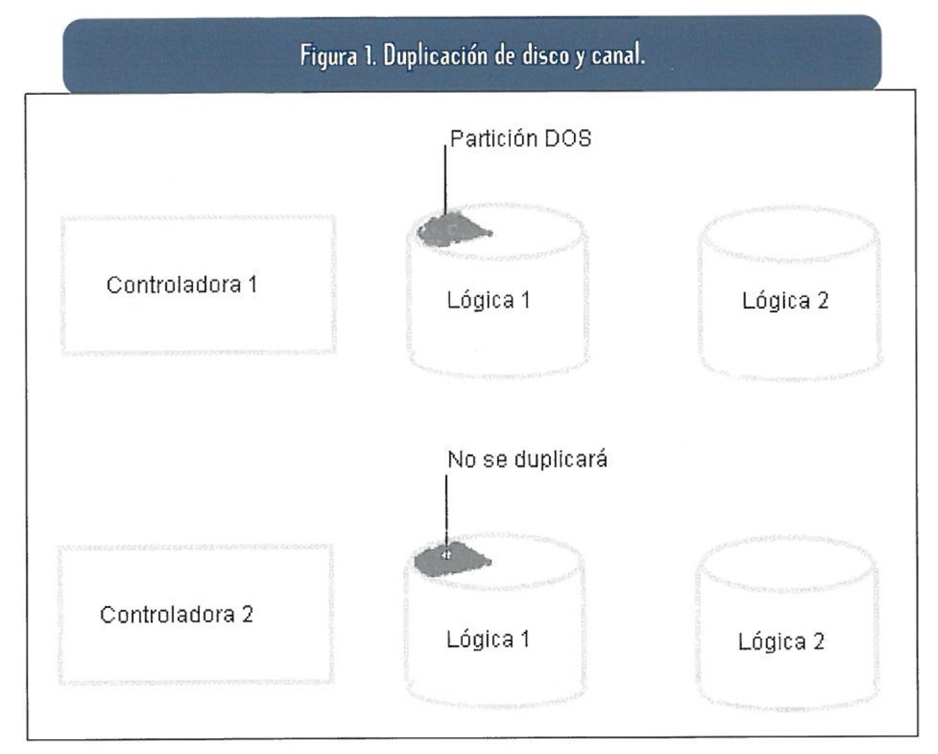
Protección de datos

La protección de datos consiste básicamente en salvaguardar los equipos informáticos frente a daños accidentales o intencionados. Estos daños incluyen fallos del *hardware*, la pérdida física de datos y el acceso a bases de datos por personas no autorizadas que podrían robar, borrar o estropear información. Algunas técnicas sencillas pueden dificultar la delincuencia informática. Por ejemplo, el acceso a información confidencial puede evitarse destruyendo la información impresa, impidiendo que otras personas puedan observar la pantalla del ordenador, manteniendo la información y los ordenadores bajo llave o retirando de las mesas los documentos sensibles. A parte de estas técnicas 'de sentido común' también tenemos a nuestra disposición otras como crear un sistema de los denominados 'tolerante a fallos'; dos o más ordenadores funcionan a la vez de manera redundante, por lo que si una parte del sistema falla el resto asume el control.

Para evitar problemas en caso de apagón eléctrico existen las denominadas UPS (acrónimo de *Uninterrupted Power Supply*) que permiten mantener el sistema informático en funcionamiento, por lo menos, el tiempo necesario para apagarlo sin pérdida de datos. Sin embargo, la única forma de garantizar la integridad física de los datos es mediante copias de seguridad.

A las utilidades del sistema para evitar accesos indeseados a la red podemos añadir otros sistemas, como las tarjetas de contraseña; son tarjetas de plástico que no pueden ser manipuladas, dotadas de un microprocesador que almacena una clave de acceso que cambia frecuentemente de forma automática. Cuando se entra en un ordenador mediante una tarjeta de acceso, el ordenador lee la clave de la tarjeta y otra clave introducida por el usuario, y las compara respectivamente con una clave idéntica a la de la tarjeta (que el ordenador genera automáticamente) y con la clave de acceso del usuario, que está almacenada en una lista confidencial. También existen ya en el mercado sistemas biométricos basados en características personales únicas como las huellas dactilares. Concretamente existe en el mercado uno capaz de leer huellas dactilares, compararlas con las que tiene almacenadas en su base de datos y contrastarlas con la cuenta y clave de acceso del usuario. En un futuro próximo es

Figura 1. Duplicación de disco y canal.



sea imprescindible esta conexión se utilizan los llamados cortafuegos, un ordenador situado entre las computadoras de una red corporativa e Internet. El cortafuegos impide a los usuarios no autorizados acceder a los ordenadores de una red y garantiza que la información recibida de una fuente externa no contenga virus.

También se pueden emplear unos ordenadores especiales denominados servidores de seguridad que proporcionan conexiones seguras entre los ordenadores conectados en red y los sistemas externos como instalaciones de almacenamiento de datos o de impresión. Estos ordenadores de seguridad emplean el cifrado en el proceso de diálogo inicial, lo que evita una conexión entre dos ordenadores a no ser que cada uno de ellos reciba confirmación de la identidad del otro.

Existen sistemas de seguridad biométricos basados en características personales únicas, como las huellas dactilares

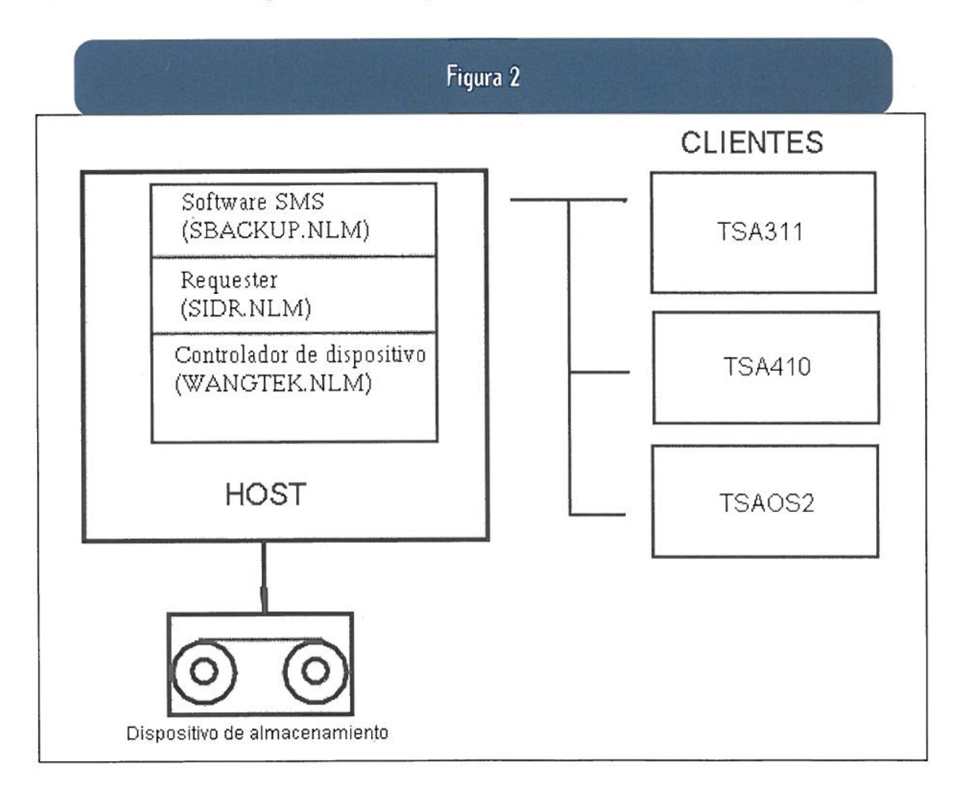
posible que estas capacidades se amplíen a los capilares de la retina, las secreciones de la piel, el ácido desoxirribonucleico (ADN) o las variaciones de la voz.

Otro peligro potencial lo constituyen los *hackers* que por su elevado nivel de conocimientos técnicos son capaces de superar determinadas medidas de protección. Su motivación abarca desde el espionaje industrial hasta el mero desafío personal. Frente a ellos poco podemos hacer. Recientemente un *hacker* entró 'a trasto' en el sistema informático del Pentágono, reventó más de 200 claves y accedió a información secreta. Internet, con sus grandes facilidades de conectividad, permite a un usuario experto intentar el acceso remoto de forma anónima a cualquier máquina conectada. Sin embargo, las redes corporativas u ordenadores con datos confidenciales no suelen estar conectadas a Internet; y en el caso de que

Proteger la red

En ocasiones, una perfecta protección de los datos, del contenido, se nos va al traste por descuidar el continente. Podemos tener mil claves de seguridad

Figura 2



que de nada nos sirven si nos roban el propio servidor para venderlo por piezas. O un incendio puede estropearnos el servidor junto con las copias de respaldo del sistema que se hallaban cerca de éste. Para evitar cosas como éstas lo más usual es poner el servidor en un cuarto aparte que se pueda cerrar bajo llave; a ser posible situado en un lugar con mucho tránsito de personas. Las copias de seguridad se guardarán lejos del servidor. Si todo este tinglado nos resulta excesivo podemos fijar el chasis del servidor al suelo o a una mesa —mejor que sea a una mesa, por si una cañería rota nos inunda el suelo— . Podemos añadir también una carcasa que disponga de algún sistema de refrigeración o ventilación y fijarla.

Respecto al fuego, es muy importante que el sistema contra incendios no tire agua sobre agua sobre los ordenadores o proteger éstos ante tal eventualidad. Los extintores de la sala han de ser de dióxido de carbono ya que son adecuados para la extinción de fuegos eléctricos y no dejan residuos que puedan dañar el equipo después de apagar el incendio.

En ocasiones, entrar a una red es tan sencillo como sentarse ante un terminal y ponerse a teclear. La razón: el usuario no se desconectó. Esta situación puede evitarse asegurándose de que el usuario se desconecta. Para ello son muy útiles las restricciones de estación y de tiempo de conexión al servidor. De esta manera el servidor cancelará la conexión si algún usuario olvida desconectarse de la red al acabar su jornada laboral. Respecto a este punto es muy importante dar a los usuarios una formación adecuada. Hay que enseñarles a conectarse y desconectarse adecuadamente, a proteger su contraseña (si es preciso, amenazar con fusilar al amanecer al que la apunte debajo del teclado) y a utilizar protectores de pantalla protegidos con contraseña si dejan desatendida su estación durante un tiempo, por ejemplo durante las comidas.

Otra posible causa de pérdida de datos es la alimentación eléctrica. En raras ocasiones las compañías eléctricas nos suministran la energía eléctrica en una onda estable, ya que está contaminada por caídas y picos de tensión y los equipos informáticos pueden reaccionar de forma impredecible ante estas irregularidades en la corriente eléctrica. Por ejemplo, puede suceder que la memoria cambie de estado y se corrompan datos o que aparezcan errores de protección general y errores de interrupciones no enmascarables. Estos errores son achacados a fallos del programa, pero a menudo son producidos por el ruido eléctrico. Los equipos permanentemente sometidos a estos picos de tensión pueden sufrir lo que se llama 'muerte lenta'. Tras un tiempo de sufrir irregularidades en la tensión eléctrica los chips degeneran y empiezan a fallar de forma sutil.

Si el pico es muy grande puede dañarnos el equipo de forma irremisible. No hace mucho, en Madrid, la reparación de un transformador de la red eléctrica provocó que la tensión que llegaba a comercios y hogares pasara de los 220 voltios habituales a 400 voltios. Naturalmente arrasó todo lo que se hallase conectado a la red eléctrica en ese momento.

A veces para entrar a una red sólo hace falta sentarse ante un terminal y ponerse a teclear

Para evitar este tipo de problemas es conveniente utilizar estabilizadores que eliminen los picos de tensión y suministren la energía de forma continua y estable.

Los virus

Los virus constituyen uno de los mayores peligros para las redes. Establecer procedimientos rutinarios para detectarlos y prevenirlos debe formar parte de nuestra estrategia general de seguridad en la red.

Los virus se extienden modificando ficheros y su capacidad de infectarlos viene determinada por los derechos potenciales del usuario cuya máquina está infectada. Existen numerosos programas antivirus especialmente diseñados para localizar virus en redes, sin embargo, la prevención constituye el mejor sistema para evitar la introducción de virus en la red. No tener un sistema de prevención es tanto como jugar a la ruleta rusa. Por muy bueno que sea nuestro programa antivirus, siempre cabe la posibilidad de que alguno pase inadvertido, ya que aparece un nuevo virus casi cada día.

Teóricamente los virus pueden evitarse cerrando los puntos de acceso a la red. Estos puntos de acceso son, principalmente, las disqueteras y las conexiones por módem. Así pues los disquetes que puedan traer los usuarios y el *software* traído desde las BBSs o Internet deben ser comprobados o prohibidos. En este sentido lo mejor es utilizar estaciones de trabajo sin disqueteras ni unidad de CD ROM. El *software* traído por los usuarios suele ser la mayor fuente de entrada de virus a la red. Esto sin contar que muchos usuarios utilizan la red para piratear más y mejor, lo cual nos puede ocasionar un serio disgusto si somos objeto de una inspección.

Para conseguir una buena protección de nuestros datos y programas frente a los virus podemos tomar las siguientes medidas:

- Limitar el número de estaciones desde las que se puede entrar como administrador del sistema, quien deberá entrar a la red como tal únicamente cuando sea estrictamente necesario. Del mismo modo, se debe procurar que las cuentas con equivalencias de seguridad de administrador sean las menos posibles. De esta manera evitaremos que el virus tenga tanto poder (en este caso destructivo) como el administrador del sistema.

• Restringir los derechos de los usuarios sobre sus directorios personales, en especial, no darles derechos de supervisor y control de acceso sobre sus directorios.

• Cada vez que instalemos nuevos programas, preocuparnos por marcar los ficheros ejecutables como de sólo lectura'. Esto evitará la diseminación de los posibles virus.

• Controlar estrictamente el uso que se hace del módem en la red.

• Pasar regularmente un programa antivirus. Es conveniente tener al menos dos programas distintos y actualizarlos regularmente.

• Concienciar a los usuarios del peligro que supone el hecho de traer programas no comprobados y restringir o controlar en lo posible estas prácticas. Si no podemos evitar —normalmente no se puede— que los usuarios utilicen la red para distribuir el último salvapantallas graciosísimo que se han traído de Internet, debemos exigirles que al menos se lo den a algún administrador de la red para que le pase un antivirus.

• Utilizar *software* original y, aún así, pasarle siempre un antivirus. Respecto al software que instalemos, y para proteger nuestra inversión, lo primero que debemos hacer es una copia de seguridad y trabajar siempre con ésta. Así no castigaremos los disquetes originales con las instalaciones que podamos realizar.

• Realizar copias de seguridad del sistema después de haber pasado un antivirus y guardarlas durante un tiempo prudencial. De este modo si descubrimos un virus que ha estado oculto durante algunas semanas, podremos restaurar un copia de seguridad libre de virus.

Hay que hacer especial mención a un tipo de *troyanos* que aparentan ser una utilidad para mejorar el *login*. Lo que hacen en realidad es recolectar los nombres de usuario y sus contraseñas, que luego serán utilizadas para entrar a la red.

Técnicas de tolerancia frente a fallos

Hasta ahora hemos visto modos de proteger nuestros datos que son prácticamente gratis. Las técnicas de tolerancia frente a fallos son caras y hay que evaluar el coste de una eventual pérdida de los datos frente al coste de protegerlos. Pongamos por ejemplo una empresa en la que existen dos turnos de trabajo de ocho horas. Si decidimos no aplicar técnicas de tolerancia frente a fallos y realizar copias de seguridad una vez al día, tendremos dieciséis horas de trabajo al descubierto. Como administradores, debemos informar a la empresa de este riesgo potencial para que evalúe el coste de perder esas horas de trabajo frente al coste de, por ejemplo, duplicar los discos. Otra solución obvia es aumentar la frecuencia de las copias de seguridad hasta que el nivel de riesgo sea aceptable por la empresa, aunque esto se realizará a costa del rendimiento del servidor y, por tanto, de una eventual caída de la productividad.

En general, las técnicas de tolerancia frente a fallos sólo se aplican en casos en los que, por lo delicado o costoso de la información que se maneja, no se puede perder un sólo *byte* de información. Existen sistemas que, sencillamente, no pueden pararse nunca. Tal es el caso de los bancos, para los que las consideraciones sobre costes son bizantinas, ya que lo que está en juego es la propia supervivencia de la empresa y se destinan los recursos que hagan falta.

Para montar un sistema tolerante a fallos podemos duplicar los discos o duplicar los servidores. Veamos cómo: la duplicación de discos nos ofrece protección frente a los fallos en disco y consiste en duplicar los datos continuamente desde el disco principal al secundario de forma que si falla la unidad principal, la secundaria asumirá su lugar hasta que reemplacemos la unidad defectuosa. Este método no nos ofrece protección contra los fallos en la tarjeta controladora, por lo que resulta recomendable aplicar también la duplicación del canal, esto es, duplicar tanto los discos como las tarjetas. Duplicar el canal también resulta muy útil cuando se tiene un volumen repartido en varios discos.

Supongamos que tenemos controladoras con dos discos cada una. Uno de los discos tendría una partición física 0 para el DOS y el resto de las particiones irían del 1 al 4. Netware asignará números de partición lógica a las unidades duplicadas de modo que cada disco y su gemelo serán vistos como una sola partición lógica. Es decir, donde tenemos cinco particiones físicas (dos del disco con la partición DOS y las tres restantes del resto de los discos), Netware montará dos particiones lógicas —una para cada disco y su gemelo—. En cuanto a las particiones físicas, donde antes teníamos cinco, ahora sólo tendremos cuatro ya que la partición DOS de uno de los discos no será duplicada en su gemelo y el espacio que ocupaba quedará sin usar.

La duplicación de servidores se implementa en Netware mediante Novell Netware SFT III. Es el nivel más alto de tolerancia a fallos que ofrece Novell, viene con las versiones 4, pero se necesita una licencia especial para instalarlo. La disponibilidad de datos y aplicaciones se asegura duplicando la memoria y la información de los volúmenes en otro servidor gemelo. La sincronización de ambos servidores se realiza en tiempo real y, si falla el servidor primario, el servidor gemelo asumirá inmediatamente sus funciones. Los usuarios ni siquiera notarían que el servidor principal ha dejado de funcionar. De este modo la información y aplicaciones están siempre disponibles.

Cuando, tras reparar el servidor principal, lo reconectamos a la red, ambos servidores se sincronizarán automáticamente.

Para implementar este nivel de seguridad necesitaremos:

- Dos servidores idénticos, con la misma cantidad de memoria RAM e idéntica capacidad de almacenamiento en disco.

- Un adaptador de comunicaciones para el enlace de servidor duplicado (MSL *Mirrored Server Links*). Nos lo puede proporcionar la misma casa fabricante de la tarjeta de red.

- Tarjetas de red idénticas para cada servidor.

- La misma versión de DOS con la que arranca el servidor e idénticas versiones de todos los controladores que utilicemos.

SFT III nos proporcionará, además de la tolerancia frente a fallos, la capacidad de realizar operaciones de mantenimiento en el servidor sin que por ello se vea afectada la funcionalidad de la red.

Copias de seguridad

Realizar copias de seguridad del sistema es la manera de proteger datos que mejor relación precio/prestaciones nos ofrece. Para tener un buen sistema de copias de seguridad lo primero que tenemos que hacer es una copia de respaldo de todo el sistema. Esto hay que hacerlo regularmente; también cuando realicemos cambios importantes en la estructura de directorios, el software o la configuración. Con esta copia de respaldo en la mano realizaremos con la frecuencia que estimemos necesaria (usualmente a diario), copias de seguridad incrementales para hacer copia de los archivos modificados o añadidos desde la última copia de seguridad. También debemos planificar las copias de seguridad durante las horas en las que menos usuarios trabajen, ya que no se copiarán los archivos que están abiertos en el momento de realizar la copia de respaldo. Como comentamos anteriormente, es conveniente guardar las copias de seguridad lejos del servidor, preferiblemente en un edificio distinto. Igualmente es muy importante efectuar pruebas de restauración regulares. De esta manera comprobaremos la bondad del software y hardware que utilicemos, que las copias se realizaron correctamente y que la cinta está en buenas condiciones. Reemplazaremos la cinta a la menor sospecha sobre su estado de conservación; el sólo hecho de sospechar de una cinta es motivo más que suficiente para reemplazarla. En informática no hay que renunciar a la intuición.

Hay que asegurarse también de que se realiza copia de seguridad del árbol NDS, ya que allí se encuentran las cuentas de usuario y sus contraseñas. Si tenemos una red con varios servidores es difícil que se estropee esta base de datos, ya que en las versiones 4 de Netware la copia de seguridad de la base de datos NDS se hace duplicándola al menos en dos servidores. Aún así es conveniente tener copia de respaldo, ya que si desaparecieran o se corrompieran todas las réplicas podríamos restaurarla y el sistema operativo reconstruiría todas las réplicas a partir de la restaurada.

Para realizar copias de seguridad incrementales, existe un método rotatorio muy popular llamado 'de las 20 cintas' o 'del abuelo'. Para poner en práctica este método seon necesarias veinte cintas que se dividen en varios grupos, el de diario, el semanal y el mensual, de la siguiente forma:

- Cuatro cintas para el grupo diario rotuladas lunes, martes, miércoles y jueves.

- Cuatro cintas para el grupo semanal rotuladas viernes1, viernes2, viernes3 y viernes4 para utilizar al final de la semana.

- Doce cintas para el grupo mensual rotuladas con los nombres de los meses para utilizar al final de cada mes.

Tal como se describe, este método está pensado para semanas laborales de cinco días. Si tenemos más días laborales debemos incrementar el número de cintas para diario.

Un punto muy importante a considerar, y que requiere una actuación inmediata, es el mensaje del servidor *"Redirection Blocks Low"* (bloques de redirección bajos). Los bloques de redirección son un espacio (normalmente un 2% de la capacidad del disco) que se reserva en el disco durante la instalación del servidor para redireccionar datos que se han intentado grabar en un sector defectuoso del disco. Si el SO detecta un sector en mal estado al grabar información en el disco, marcará este sector como defectuoso y grabará los datos en el espacio que hemos reservado para este fin. El mensaje *"Redirection Blocks Low"* indica que el disco duro está a punto de estropearse; falla tanto que apenas queda espacio para redireccionar los bloques defectuosos: el disco se nos muere. Frente a esto debemos:

- Avisar a los usuarios para que salven su trabajo a un disco local y salgan de la red.

- Realizar inmediatamente una copia de seguridad.

- Bajar el servidor.

- Reiniciar el disco y pasarle todos los tests habidos y por haber para determinar su estado. En cualquier caso, lo mejor es sustituir el disco cuanto antes, ya que podría fallar definitivamente algún tiempo después.

Servicios de gestión de almacenamiento

Los servicios de gestión de almacenamiento (de las siglas SMS, *Storage Management Services*) son un conjunto de

El mensaje Redirection Blocks Low indica que el disco duro está a punto de estropearse

Figura 3

```
Session Date and Time: 16/04/97 09:11:29
Description: span a tape
Target: TRAIN (NetWare v3.11)
Media Set ID: train #1 16/04/97 09:10:29
Backed up: File Server

16/04/97 09.12:30 Successfully Backed up 1 parents and 10 children, total written:
821.0.kb, elapsed time 00:00:42!
*********************The following errors occurred in TSA*****************************

16/04/97 09:20:00 Trustee "FRED" was not restored for "SYS:PUBLIC/FILER.EXE"

********************The following data sets were not backed up!*********************

SYS:X.LSB (In use)

10/10/97 08:00:00 Restore Session Begin
```

especificaciones de Novell para realizar copias de seguridad en entornos de red. Básicamente, dichas especificaciones son las siguientes:

- Permitir copias de seguridad completas, incrementales, diferenciales y personalizadas.

- Permitir la copia de archivos de DOS, Macintosh, NFS y OS/2, manteniendo los atributos propios de cada tipo de archivo.

- Restaurar las copias de seguridad manteniendo su posición original o bien en otra.

- Permitir realizar copia de seguridad no sólo del servidor, sino de toda la red.

- Ofrecer compatibilidad hacia atrás con antiguas copias de seguridad, de modo que si el software SMS cambia, las antiguas copias de seguridad sean accesibles con el nuevo software.

Antes de explicar la forma en que funciona el software SMS debemos asumir la terminología inherente a este estándar.

Host es un servidor que tiene conectado un dispositivo de almacenamiento que ejecuta el software SMS y realiza la copia de seguridad de los dispositivos *destino*. Cuando hace copia de seguridad de sus propios datos es tanto *host* como *destino.*

Destino sería cualquier nodo de la red (ya sea servidor o estación de trabajo) que es objeto de la copia de seguridad por parte del *host.*

TSA (Target Service Agent, Agente de servicio de seguridad) es un programa auxiliar que ayuda a gestionar la transferencia de datos entre el *host* y el *destino.*

Conjunto de datos son todos aquellos datos que pueden ser manipulados por el software SMS.

SIDF (System Independent Data Format, Formato de datos independiente del sistema) es el formato de datos en que se almacenan las copias de seguridad. El TSA es el que se ocupa de suministrar los datos en formato SIDF al dispositivo de copia de seguridad SMS.

Novell ofrece SBACKUP.NLM (módulo cargable de Novell) como software SMS y ofrece TSAs para servidores Novell Netware en sus versiones 3 y 4, para la base de datos del árbol NDS y para estaciones de trabajo DOS y OS/2.

Como se muestra en la figura 2 un sistema básico SMS consta de un servidor que soporta SBACKUP.NLM y uno o varios agentes de servicio de destino o TSAs. Uno de los objetivos del software SMS es que el hardware y el software se integren tanto como sea posible. De este modo los fabricantes de dispositivos pueden crear controladores específicos que permitan integrar su hardware con SMS y así posibilitar que el software SMS no se ocupe del tipo de dispositivo que tiene conectado, ya que esa parte la gestiona el módulo del fabricante de hardware.

Cómo hacer una copia de seguridad

Para realizar una copia de seguridad o restaurar archivos tendremos que realizar las siguientes tareas:

- cargar los TSAs en el *destino* mediante el mandato:
  LOAD TSA.
  Este programa cargará todos los módulos necesarios. En el *host* cargaremos el módulo SBACKUP. Para ello teclearemos:
  LOAD SBACKUP
  Se nos pedirá un nombre de usuario y clave de acceso.

- Elegir el destino de la copia de seguridad o restauración. Tras haber dado el nombre de usuario y clave de acceso se nos preguntará sobre qué dispositivo (si tenemos más de uno) queremos restaurar o copiar. Tras

seleccionar el controlador adecuado aparecerá el menú principal. Elegiremos la opción *Select Target to Backup/Restore* (Seleccionar destino de copia/restauración). Si hay más de un TSA se nos presentará una lista de todos los *destinos* posibles donde elegiremos aquellos que nos interesen. También deberemos dar un nombre de usuario y clave de acceso para los destinos.

SBACKUP no realizará copia de los archivos que estén en uso durante el proceso de Backup

• Realizar la copia de seguridad. Para copiar los datos a la cinta seleccionaremos la opción *Backup Menu* del menú principal, lo que nos llevará las opciones de copia de seguridad. Seleccionaremos el directorio de trabajo, que ha de estar siempre en el servidor *host,* con la opción *Select Working directory* . Aparecerá una ventana en la que podemos teclear la ruta de acceso o pulsar la tecla insert para que aparezca una lista de directorios. Si especificamos un directorio que no existe el programa nos pedirá confirmación para crearlo. Una vez especificado el directorio de trabajo seleccionamos la opción *Backup Selected Target*. Aparecerá una ventana donde podemos establecer algunas opciones de *Backup.* Presionaremos la tecla escape (Esc) dos veces para guardar las opciones. Finalmente el sistema nos preguntará si queremos realizar la copia de seguridad ahora mismo o más tarde. En el caso de elegir más tarde tendremos que introducir la fecha y hora.

Cuando comience la copia podremos ver una ventana de estado donde se

Figura 4

```
Session Date and Time: 10/08/97 11:50:14 Log
Description:
Target TRAIN (NetWare v3.11)
Media Set ID: xx #1 10/08/97 11:50:14
Backed up SYS:PUBLIC
001 00000001  00000048 SYS:PUBLIC/
001 00000001  000001C8 SKLDIBI.HLP
001 00000001  00008D68 UPGRADE.HLP
001 00000001  00013E60 SALVAGE.HLP
```

muestran los ficheros que están siendo objeto de copia, el tiempo transcurrido y los errores ocurridos durante el proceso de copia.

• Ver el registro de errores. El registro de errores (figura 3) anota todos los errores ocurridos durante una sesión de copia de seguridad. Los errores ocurridos durante el restaurado de los archivos se añaden al archivo de registro de copia de seguridad. Para ver el registro de errores elegiremos *Back up Menu* o *Restore Menu* desde el menú principal de SBACKUP y seleccionamos *View Error Log* (ver registro de errores). Aparecerá una lista de sesiones desde la que podemos elegir aquella que nos interese.

El registro de copia de seguridad (figura 4) lista todos los archivos que han sido objeto de copia, así como el número de identificación de la cinta (variará si hemos utilizado más de una). Para ver el registro de copia de seguridad elegiremos *Back up Menu* desde el menú principal. Una vez allí seleccionamos *View Back up Log* (Ver registro de copia de seguridad); aparecerá una lista de sesiones desde la cual podremos ver el registro que nos interese.

Restaurar archivos

Antes de restaurar una copia de seguridad debemos asegurarnos de que disponemos de espacio suficiente en disco. En general, debemos tener un veinte por ciento más del espacio requerido para proceder al restaurado de las copias de seguridad. Las técnicas para proceder al restaurado son prácticamente idénticas a las de copia de seguridad, por lo que no vamos a explicarlas. Tan sólo reseñar que podemos restaurar o no todos los derechos, atributos y datos tal como estaban en el momento de realizar la copia de seguridad; restaurarlos en su ubicación de origen o elegir un volumen o una estructura de directorios diferentes.

El rendimiento de SBACKUP varía dependiendo de la configuración y de si estamos haciendo copia del propio servidor o de un servidor de archivos remoto. En el primer caso la copia será unas cuatro veces más rápida que si lo hacemos de un servidor remoto.

El rendimiento de SBACKUP varía según la configuración

Para incrementar el rendimiento de SBACKUP podemos añadir parámetros a la orden LOAD SBACKUP para cambiar el número o el tamaño de los búferes para caché. La documentación de los controladores que estemos utilizando nos dará sus recomendaciones para variar estos parámetros.

Si el controlador que utilizamos necesita poca memoria los búferes serán asignados automáticamente; pero si obtenemos un mensaje del tipo *Out of Memory* debemos utilizar el comando SET RESERVED BUFFERS BELOW 16MB para aumentar el número disponible de búferes en memoria baja. Esto sólo lo podremos indicar en el archivo STARTUP.NCF.

Debemos ser muy cuidadosos si cambiamos el número o tamaño de los búferes, ya que si especificamos una combinación que exceda los 16 Mb de memoria, los datos que se graben en la cinta serán defectuosos.

Programación X-Window: Driver SVGA para XFree86 (II)

LINUX

Jorge F. Delgado Mendoza

Tras mucho trabajo, horas de depuración, ideas brillantes a mitad de noche que te impiden dormir, etc., ¡por fin has conseguido tu objetivo!: ya tienes un *driver* SVGA para tu nueva tarjeta, capaz de soportar altas resoluciones con 8 bpp. Has archivado tu copia del *Sólo Programadores* del mes pasado e, incluso, te has dado unas palmaditas en la espalda, al ver que no era el toro tan fiero como lo pintaban.

Sin embargo, no estás satisfecho del todo con el rendimiento de tu tarjeta. En MS-Windows eres capaz de usar 1024x768 16 bpp con lo que la paleta no se te queda al tresbolillo cada vez que cambias del *Netscape* a tu programa de manipulación de imágenes preferido. Además, la tarjeta parece mucho más lenta en *Linux* que en las Windows y te preguntas: ¿cómo es esto posible?, ¿no era *Linux* mucho más rápido en todo?, ¿no es un sistema puro de 32 bits?, ¿qué es lo que sucede?

En este artículo no sólo vamos a dar respuesta a todas estas preguntas, sino que iremos más allá, describiendo de una manera concisa cuáles son los pasos que hay que dar para soportar más colores, obtener más funcionalidad y, finalmente, alcanzar la mayor velocidad posible en los gráficos de una placa de vídeo.

En el primer artículo de la serie dimos soporte para una tarjeta de vídeo nueva al servidor SVGA de XFree86. Aunque la elaboración fue larga y compleja, lo que hicimos no fue más que un rudimentario comienzo: apenas hemos tocado los aspectos más importantes de la programación de los aceleradores gráficos actuales. Sin embargo, sentamos una base a partir de la cual podemos construir un *driver* de calidad, capaz de satisfacer al usuario más exigente y comparable, en todos los aspectos, a los que están disponibles comercialmente.

Requisitos

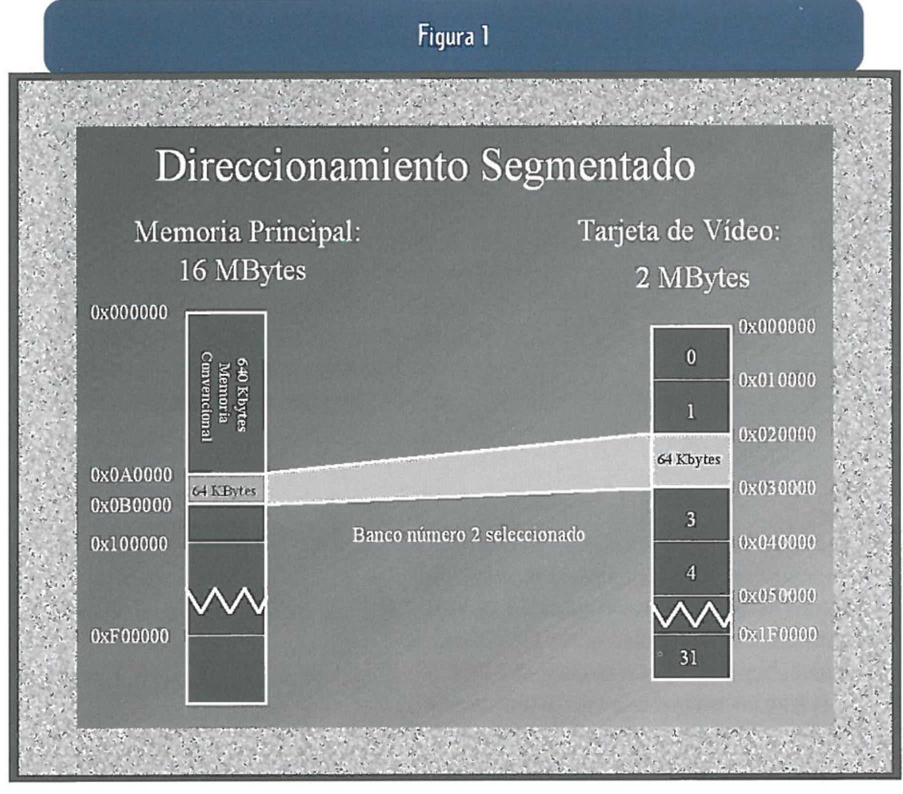
A pesar de todo, no debemos olvidar que llevar a los componentes más allá de sus especificaciones máximas, e incluso cerca de ellas, puede acarrear fatales consecuencias. A pesar del éxito obtenido tras la implementación del *driver* básico, no nos dejemos llevar por el orgullo y tomemos las precauciones adecuadas. Recuerda, un monitor quemado y una tarjeta inservible son, con frecuencia, las recompensas del programador inconsciente.

Como en el artículo anterior, lo primero que haremos será una lista de los requisitos, tanto de *software* como de conocimientos, que se deben cumplir para poder seguir adelante en la construcción del *driver*.

• **Conocimientos:** de manera similar al artículo del número de Sólo Programadores anterior, es necesario y

En el primer artículo de la serie añadimos soporte para una tarjeta de vídeo en 256 colores. En este segundo artículo, utilizaremos una nueva tarjeta para ilustrar el método que se emplea para añadir drivers HiColor y TrueColor al servidor del X-Window System. Además, aprenderemos a pasar parámetros a través del fichero de configuración del servidor.

Figura 1



conveniente dominar el lenguaje de programación C para poder continuar con el trabajo de implementación del *driver*. Además, unas nociones de ensamblador para x86, algunos conocimientos acerca del funcionamiento interno de una tarjeta de vídeo VGA compatible y saber manejarse con el sistema de puertos de E/S de un PC, son deseables aunque no sean imprescindibles.

• **Software:** necesitaremos también tener instaladas las siguientes herramientas:

- *gcc* 2.7.0 o superior.
- *make*.
- *XFree86*.
- *LinkKit*.

• **Documentación:** de nuevo vamos a necesitar el manual de registros de la tarjeta de vídeo sobre la que deseemos codificar. Además sería conveniente un manual de la especificación PCI o VLB, si se pueden conseguir, así como las especificaciones del monitor que vayamos a utilizar para las pruebas.

Direccionamiento Lineal

Por motivos que veremos más adelante, la primera característica avanzada de las modernas tarjetas de vídeo que vamos a aprovechar, es el llamado Direccionamiento Lineal o *Linear Addressing Mode*.

El estándar VGA reserva 64 Kbytes en el mapa de memoria del PC para gráficos en color , situados en la dirección 0xA0000

Como ya vimos en el primer artículo de la serie, el estándar VGA reserva para gráficos en color 64 Kbytes en el mapa de memoria del PC, situados en la dirección 0xA0000. Esta cantidad es claramente insuficiente para proporcionar modos de vídeo más allá de 320x200 8bpp ó 640x480 4bpp. Debido a ello, los fabricantes optaron por considerar esa apertura de 64 Kbytes como una ventana a la memoria total instalada en la tarjeta de vídeo.

Esta organización de la memoria, realizada de modo muy diferente por cada uno de los fabricantes, provocó la aparición de los *drivers* de las diferentes tarjetas. Éstos se limitaban, en un principio, a proporcionar las rutinas de cambio de banco necesarias para que la memoria apuntada por la ventana fuera, en cada momento, la adecuada. Más tarde, con la aparición de los aceleradores gráficos, los *drivers* se utilizaron también para proporcionar una interfaz a las funciones aceleradas de la tarjeta. Pronto se vio que esta organización de la memoria era poco eficaz, apareciendo tarjetas capaces de colocar todo su *frame buffer* en una posición de memoria más allá de la RAM instalada del PC. Este modo de organizar la memoria de vídeo se denominó *Linear Addressing Mode*.

Las principales ventajas de este método de direccionamiento son dos:

- Por un lado, *se evita la sobrecarga del cambio de banco*, ahorrándose ciclos de reloj en la ejecución de las tareas gráficas. Datos experimentales revelan incrementos de hasta un 20% en la velocidad de ejecución para ciertas combinaciones resolución/profundidad/tarjeta/sistema.
- Por otro lado, al no tener que partir ninguna línea horizontal por motivos de cambio de banco, se pueden realizar *nuevas optimizaciones* en los aceleradores con el consiguiente aumento en el rendimiento.

Existen también varias desventajas, que enumeramos a continuación:

- Es necesario soportar los dos sistemas: lineal y segmentado, por compatibilidad con las VGAs clásicas,

con el consiguiente aumento en la complejidad del diseño.

- La programación de *drivers* se vuelve más compleja, ya que es necesario realizar operaciones para ambos tipos de direccionamientos. Además las rutinas de detección han de ser más precisas, ya que la apertura lineal no debe solaparse con la memoria principal del PC.
- Los microprocesadores modernos implementan tecnologías de acceso a memoria muy avanzadas, las cuales suelen estar disponibles solo en el espacio de la memoria principal, a no ser que programemos el *micro* para abarcar más allá. De este modo podríamos encontrarnos con que, a pesar de las aceleraciones, es más rápido utilizar 0xA0000 que en un lugar fuera del rango de estas optimizaciones.

A pesar de que las desventajas parecen importantes, no lo son tanto. La primera de ellas la resuelven automáticamente los fabricantes. La segunda nos incumbe, pero veremos como la arquitectura de XFree86 hace relativamente sencillo añadir direccionamiento lineal para un *chipset* determinado. La tercera no es muy relevante ya que, hoy en día, existen programas, verdaderos *drivers* de los microprocesadores, capaces de extender los beneficios de los nuevos métodos de direccionamiento más allá de la memoria convencional instalada.

Para implementar las nuevas funciones, vamos a utilizar una nueva tarjeta imaginaria que *Solo Programadores Tech. Inc.* acaba de sacar al mercado. Esta tarjeta llamada *Accelerated'r'Us* permite modos de 16 y 24 bpp, Direccionamiento Lineal, Cursor Hardware y un rudimentario acelerador. Como en el artículo anterior, el manual de registros se puede encontrar en el CD en los siguientes formatos: *Portable Document Format* (PDF), *PostScript* (PS) y *ASCII* (txt). Para que el *driver* reconozca el nuevo *chipset*, deberemos introducir la entrada apropiada en la función **GUIdent()**.

Como hemos visto, debemos tener la precaución de que la memoria principal y la de vídeo no se solapen al colocar la tarjeta en modo lineal. Para facilitar la tarea del programador y ofrecer versatilidad ante la gran cantidad de configuraciones posibles, la mayoría de los *chipsets* permiten colocar su mapa de memoria en distintas posiciones. A pesar de ello, es posible que todas colisionen, por lo que tendrá que ser el usuario de la máquina quien decida si quiere direccionamiento lineal o el método segmentado.

En previsión de este tipo de situaciones, existe la posibilidad de pasar parámetros al servidor de X-Window a través del fichero de configuración de las X, situado típicamente en: **/etc/XF86Config**. La sintaxis es sencilla: basta con añadir las líneas correspondientes a las opciones que deseemos en la parte dedicada a la tarjeta de la siguiente manera (la descripción de la sintaxis de XF86Config se obtiene mediante el comando 'man XF86Config'):

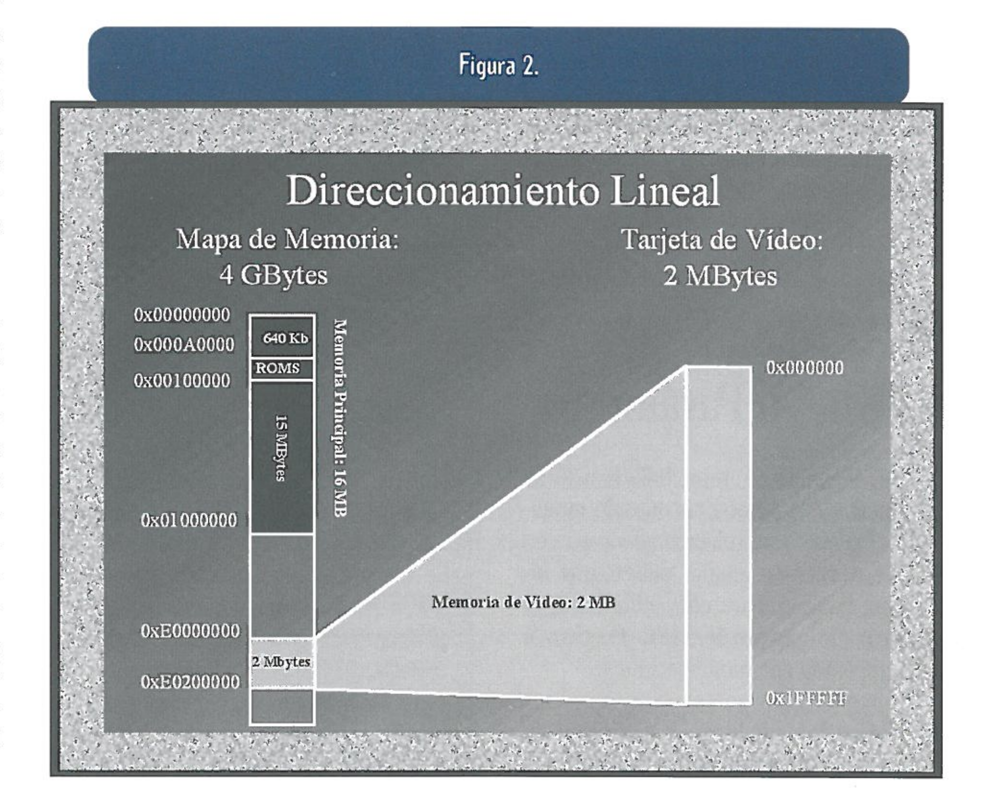
```
Option "opción1"
Option "opción2"
Option "opción3"
```

Por tanto, antes de añadir el direccionamiento lineal al servidor, vamos a aprednder cómo se pasan parámetros al *driver* de una tarjeta de vídeo.

Pasando Parámetros al Servidor

La lista de las distintas opciones que soporta el servidor SVGA se encuentra en el *LinkKit* dentro del siguiente fichero: **include/xf86_Option.h.** Toda opción que queramos añadir deberá encontrarse reflejada en él. En cuanto a la política de nombres, es preferible reutilizar opciones ya existentes cuyo efecto sea similar al nuestro, antes que añadir nuevas opciones. La razón es mantener la coherencia en la familia de *drivers* del servidor, evitando perder a los usuarios con un juego distinto de opciones por tarjeta. Por poner un ejemplo, si deseamos una opción para activar el direccionamiento lineal, es preferible reutilizar la opción existente:

Figura 2.



Option "linear" utilizada en todos los *drivers* para la citada tarea, antes que crear una nueva, del tipo: **Option "AU_linear"**. Sin embargo, es preferible añadir opciones nuevas a utilizar las ya existentes con significados distintos o que puedan llamar a equivocaciones.

La lista de las distintas opciones que soporta el servidor SVGA se encuentra en el fichero include/xf86_Option.h.

Supongamos que queremos programar la temporización de la memoria para 60ns DRAM's. En vez de utilizar la opción: **Option "clock_50"** creada para modificar el reloj del *chipset* de algunas tarjetas, lo cual no tiene ninguna relación con programar el MCLK de nuestra tarjeta de tal modo que los accesos a memoria sean gobernados por un reloj de 50 Mhz, utilizaremos una nueva del tipo: **Option "60ns_DRAM"** aunque sería mejor utilizar: **Option "med_dram",** opción que ya existe y, aunque no significa exactamente lo mismo, se puede utilizar para el mismo propósito.

Añadiendo un Nuevo Parámetro

Si después de mirar toda la lista de parámetros, no hemos encontrado ninguno que guarde una relación adecuada con lo que deseamos añadir, deberemos incluir uno nuevo. Para ello, editaremos el fichero: **include/xf86_Option.h** Introduciendo dos modificaciones:

- Miraremos la lista de *#define* del fichero, buscando los dedicados a las *Option Flags*. Una vez encontradas, añadiremos nuestra opción al final de la lista, incrementando en uno el último número de orden encontrado. Además, añadiremos un breve comentario indicando cuál es la utilidad de la opción añadida. Por ejemplo, si la última opción encontrada es:

```
#define OPTION_TGUI_PCI_WRITE_OFF 172
        /* Trident TGUI PCI burst write */
```

Añadiremos la nuestra justo después, quedando el fichero de la siguiente manera:

```
...
#define OPTION_TGUI_PCI_WRITE_OFF 172
        /* Trident TGUI PCI burst write */
#define OPTION_AU_60NS  173
        /* Accelerated'r'Us with 60ns DRAM */
...
```

- Una vez realizada esta modificación, deberemos añadir la nueva opción en la estructura de datos **OptFlagRec xf86_OptionTab[]**, que es la encargada de realizar la equivalencia entre la cadena que sigue al parámetro *Option* en el fichero de configuración y la opción adecuada.

La cadena utilizada en el fichero de configuración debería tener, como mucho, diez o doce caracteres de longitud.

```
...
{ "60n_mclk",  OPTION_AU_60NS },
...
```

Incluyendo Opciones en el Driver

Una vez tengamos todas las opciones deseadas en el fichero de configuración, bien porque vayamos a reutilizar algunas o bien porque hayamos añadido las opciones oportunas al fichero **xf86_Option.h**, podremos utilizarlas en nuestro *driver*.

Para incluir las opciones deberemos realizar tres tareas:

Listado 1.

```
#include "X.h"                  /*Headers específicos de XFree86*/
#include "input.h"
#include "screenint.h"
#include "dix.h"

#include "compiler.h"

#include "xf86.h"
#include "xf86Procs.h"
#include "xf86Priv.h"
#include "xf86_OSlib.h"
#include "xf86_HWlib.h"

#include "xf86_Config.h"        /*Headers para opciones extras*/
#include "xf86_Option.h"
#include "vga.h"
#include "region.h"
```

- Incluir los ficheros correspondientes mediante las directivas *#include* apropiadas.

- Activar las opciones que van a ser utilizadas por nuestro *driver*, de entre todas las que se encuentran en el fichero **xf86_Option.h**.

- Utilizarlas en nuestro código. Las opciones pueden servir para añadir mensajes de depuración, optimizar el comportamiento de la tarjeta basándonos en información que el usuario pueda proveer, sin que sea factible obtenerla mediante pruebas *hardware*, etc...

La primera tarea, la de incluir los ficheros adecuados, se resuelve fácilmente. Los ficheros que debemos añadir son:

include/dix.h
include/xf86Procs.h
include/xf86_Config.h
include/xf86_Option.h
include/region.h

En cuanto al orden, un listado completo de las inclusiones del *driver* se puede ver en el Listado 1.

Una vez hemos realizado las inclusiones necesarias, es necesario activar todas las opciones que vayamos a utilizar en el código del *driver*. Para ello añadiremos una serie de líneas al final del procedimiento de detección de la tarjeta: **GUProbe()**. Estas líneas, de las cuales debe haber una por opción que deseemos añadir, deben tener la siguiente estructura:

```
OFLG_SET(NOMBRE,
&CHIPSET.ChipOptionFlags);
/* Comentario */
```

En cuanto al contenido de los parámetros NOMBRE y CHIPSET, deberemos tener en cuenta los siguientes aspectos:

- NOMBRE: es el nombre de la opción tal y como está definida en el fichero de opciones: **include/xf86_Option.h**.

- CHIPSET: es la cadena de caracteres que hemos utilizado a lo largo de todo el *driver* para identificar nuestras funciones y estructuras de datos.

En el caso de nuestra tarjeta de ejemplo, para la cual vamos a ampliar el código realizado en el primer artículo, y considerando el ejemplo de opción descrito anteriormente, la línea que debemos añadir quedaría de la siguiente manera:

```
OFLG_SET(OPTION_AU_60NS,
&GU.ChipOptionFlags);
/* Opción utilizada para ajustar la
 * temporización del MCLK de la tarjeta
 * para 60ns DRAM */
```

En la elaboración del *driver* vamos a incluir varias opciones, algunas de las cuales también serán utilizadas por el antiguo *chipset Graphics'r'Us* con el fin de darle más versatilidad. Las opciones que vamos a utilizar son:

- **"fast_dram"**: esta opción se utilizará cuando la memoria instalada en la tarjeta sea capaz de soportar MCLK's de 70Mhz. Es decir, cuando tengamos 35ns EDO, SDRAM o MDRAM.

- **"med_dram"**: de modo similar a la anterior, esta opción indicará al servidor que la memoria instalada soporta 60 Mhz de MCLK.

- **"slow_dram"**: con esta opción configuraremos la tarjeta para la señal MCLK oscile con una frecuencia de 45 Mhz.

- **"slow_edodram"**: esta opción se utilizará para indicarnos que la memoria instalada en la tarjeta soporta, como mucho, 55 Mhz.

- **"fifo_conservative"**: esta opción forzará al servidor a utilizar una profundidad de la cola de comandos de doce.

- **"fifo_moderate"**: la profundidad de la FIFO se situará en seis.

- **"fifo_aggressive"**: utilizando esta opción, configuraremos la tarjeta para que ejecute comandos tal y como le llegan. Es la opción más rápida.

- **"linear"**: con esta opción habilitaremos el direccionamiento lineal de la tarjeta. En un principio, vamos a fijar la dirección del *frame buffer* en

Listado 2

```
static Bool
GUProbe()
{

...
OFLG_SET(OPTION_FAST_DRAM, &GU.ChipOptionFlags);
OFLG_SET(OPTION_MED_DRAM, &GU.ChipOptionFlags);
OFLG_SET(OPTION_SLOW_DRAM, &GU.ChipOptionFlags);
OFLG_SET(OPTION_SLOW_EDODRAM, &GU.ChipOptionFlags);
OFLG_SET(OPTION_FIFO_CONSERV, &GU.ChipOptionFlags);
OFLG_SET(OPTION_FIFO_MODERATE, &GU.ChipOptionFlags);
OFLG_SET(OPTION_FIFO_AGGRESSIVE, &GU.ChipOptionFlags);
OFLG_SET(OPTION_LINEAR, &GU.ChipOptionFlags);
OFLG_SET(OPTION_NO_WAIT, &GU.ChipOptionFlags);
OFLG_SET(OPTION_WAIT_STATE, &GU.ChipOptionFlags);
...

}
```

0xE0000000. De esta manera, podremos usar direccionamiento lineal siempre que tengamos menos de 3'75 Gbytes de RAM instalados y ninguna otra tarjeta utilice esta dirección.

- "no_wait": al colocar esta opción en el fichero de configuración, indicaremos al *driver* que debe programar la interfaz para que no existan estados de espera en el *bus*.

- "wait_state": con esta opción programaremos la tarjeta para que añada un estado de espera en las operaciones de lectura y escritura que se efectúen a través del *bus*.

Todas estas opciones influirán en la programación de las tarjetas basadas en el *chipset Accelerated'r'Us*. Sin embargo, sólo las relativas a los estados de espera y a la FIFO tendrán alguna utilidad en las tarjetas basadas en un *Graphics'r'Us*.

Todas las opciones que vamos a utilizar en el servidor están ya presentes en el fichero de opciones, por lo que no es necesario modificarlo. El código utilizado para declarar las opciones soportadas por el *driver* se puede ver en el Listado 2.

Una vez hayamos realizado todas las operaciones descritas, estaremos en condiciones de poder utilizar opciones en nuestro código. Para comprender su modo de utilización, vamos a describir, sin entrar en detalle, el proceso de arranque del servidor del *X Window System*. El proceso es un ejemplo muy sencillo que no refleja el exacto funcionamiento interno del servidor, si se desea obtener más información, se puede contactar con el autor en su dirección de correo.

Cuando realizamos la llamada al servidor de X, se realizan, además de otras, las siguientes operaciones :

- Inicialización de las estructuras de datos genéricas.

- Lectura del fichero de configuración, rellenando las estructuras de datos pertinentes, incluyendo, pero no limitándose a modos de vídeo, memoria, chipset, ratón, teclado, etc.

- Detección de la tarjeta de vídeo, para ello arranca secuencialmente todas las funciones ***Probe()** hasta que una de ellas encuentra el *chipset* presente en el sistema.

- Una vez detectado el *hardware*, inicializa los dispositivos mediante las funciones ***FbInit()**, enviando a la tarjeta el primer modo indicado en el fichero de configuración con una llamada a ***Init()** la cual se ejecuta cada vez que cambiamos de modo.

Por lo tanto, cuando llega al segundo paso encuentra las opciones que se hallen presentes en el fichero de configuración, guardándolas en una tabla de opciones activas. Una vez ha llegado a la función de detección y en el caso de que sea positiva, crea una nueva estructura de datos, conteniendo entradas para todas las opciones que soporta el *driver*. Tras ello, rellena con valores de verdad la estructura creada, en función de que la entrada esté presente también en la tabla de opciones activas, o no lo esté. En el caso de que la tabla contenga alguna opción no soportada, la traza del servidor presentará un mensaje de error.

De este proceso, lo único que nos interesa es que el servidor guarda, en alguna parte, una tabla en la que tenemos información sobre las opciones que se han pasado o dejado de pasar al *driver*. De este modo podremos utilizar una función, en realidad una macro, que va a devolver un valor de *Verdadero* si la opción se encontró en el fichero de configuración y falso en caso contrario. La función que debemos utilizar es:

```
Bool OFLG_ISSET(ARG1,
                        &vga256InfoRec.options)
```

Donde el valor de ARG1 es el nombre de la opción por cuya presencia en el fichero de configuración nos interesamos. Generalmente se emplea unida a sentencias *if () else*, del siguiente modo:

```
if (OFLG_ISSET(OPTION_LINEAR,
                        &vga256InfoRec.options))
{       /* Código que configura el frame buffer como
        lineal */
}else
{       /* Código del direccionamiento segmentado
        */
}
```

En este ejemplo preguntamos al servidor si el usuario ha especificado que desea utilizar el direccionamiento lineal en el fichero de configuración y, en función de la respuesta, realizamos unas operaciones u otras.

Un Inciso: La Traza del Servidor

Cuando arranca el servidor, envía a la consola desde la que lo hayamos ejecutado una traza de ejecución, donde informa de las acciones que va realizando.

Al arrancar el servidor, envía a la consola desde la que lo hayamos ejecutado una traza de ejecución

Fundamentalmente informa del *chipset* que ha detectado, de los modos que es capaz de ofrecer, de las *Fonts* que añade, etc. Nosotros, hasta ahora, no hemos añadido mensajes a esa traza, pero conviene hacerlo en las fases de codificación y depuración. Así obtendremos información sobre lo que se está haciendo y podremos ver si es correcto. Además, de esta manera, si tenemos gente probando el código realizado, será suficiente con que nos envíen la traza del servidor para poder realizar modificaciones.

La función utilizada para enviar mensajes a la traza es: *ERRORF(*CADENA*);* donde CADENA es un tipo *(char *)* con una estructura similar a la de la función estándar *printf*(), pero en la que han de estar presentes dos argumentos:

```
"%s %s: <chipset>: información deseada. \n",
    FLAG1,FLAG2
```

En cuanto a los argumentos presentes en la cadena, tenemos:

- *<chipset>:* indica qué *driver* está produciendo la traza. En nuestro caso, *"SoloProgramadores"*.
- *información deseada:* aquí pondremos la información que deseemos presentar en la traza.
- *FLAG1:* puede tomar dos valores. Si el mensaje es debido a algo que hemos detectado internamente en el *driver* debemos utilizar XCONFIG_PROBED. En caso contrario, si el mensaje es relativo a parámetros definidos en el fichero de configuración, utilizaremos:
  XCONFIG_GIVEN.

La función que se utiliza para enviar mensajes a la traza es: ERRORF(CADENA)

- *FLAG2:* es una cadena que determina qué servidor se está utilizando en cada caso determinado. En este campo deberemos indicar lo siguiente: **vga256InfoRec.name**.

Como ejemplo, vamos a suponer que acabamos de colocar la tarjeta en modo de direccionamiento lineal, a petición del usuario a través de **XF86Config**. La traza podría quedar de la siguiente manera:

```
ERRORF("%s %s: SoloProgramadores: Entrando
    en Direccionamiento Lineal\n",
    XCONFIG_GIVEN,vga256InfoRec.name);
```

Una última palabra: es muy útil utilizar trazas de cada operación que realicemos, ya que evitan muchos quebraderos de cabeza. Además, un servidor que informe de lo que está haciendo suele dejar muy tranquilo al usuario, ya que contribuye a dar la impresión de estar usando un producto de calidad.

En el caso de que estemos depurando, conviene añadir un *fflush()* justo después de *ERRORF* para que imprima en pantalla inmediatamente, sin ejecutar más operaciones ni esperar a que se llene el *buffer* de impresión.

Listado 3

```
static void
GUProbe()
{

...
if (
        (OFLG_ISSET(OPTION_LINEAR, &vga256InfoRec.options) &&
        (SP_chipset == AU)
   )
        {
        GU.ChipUseLinearAddressing = TRUE;
        GU.ChipLinearBase = 0xE00000;
                /* Por Defecto en el Mbyte 14 */
        if (SP_bus == PCI)
                {
                GU.ChipLinearBase = 0xE0000000;
                /* A no ser que tengamos PCI, entonces en 3'75 GB */
                }
        GU.ChipLinearSize = vga256InfoRec.videoRam;
                /* Tamaño = Memoria Detectada */
        }
...

}
```

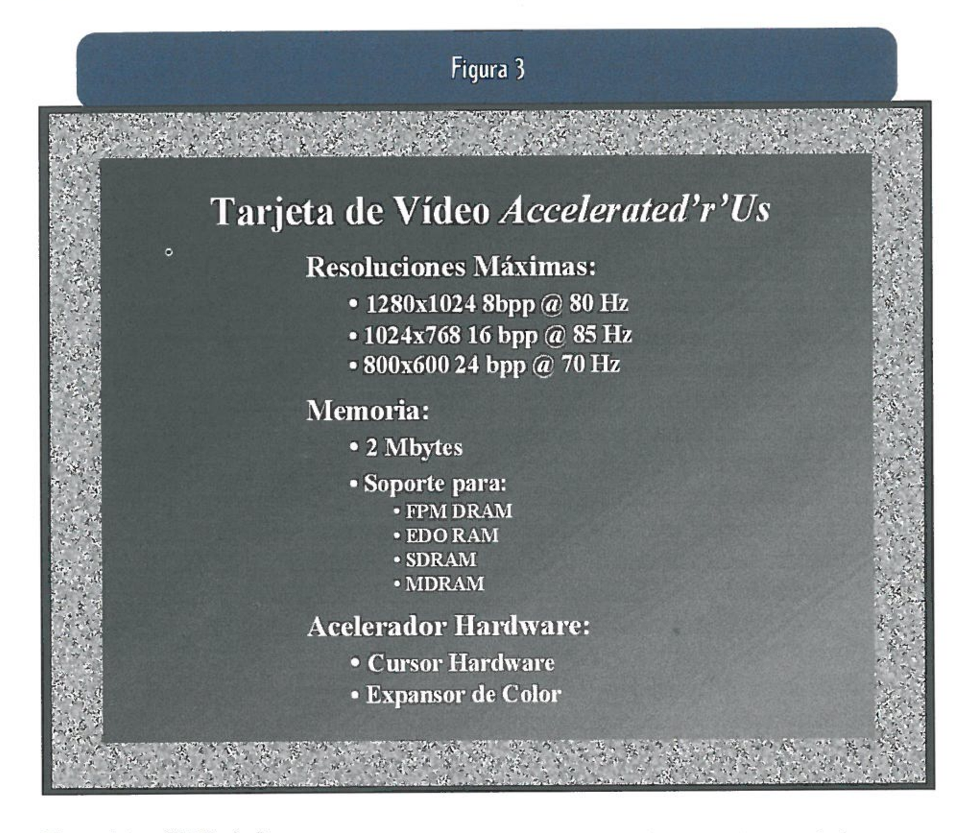
Ahora Sí: Memoria Lineal

Ahora que podemos seleccionar el modo de direccionamiento, es el momento de implementar la apertura lineal. Realizaremos la secuencia de operaciones siguiente:

Función GUProbe():

- Comprobaremos si el usuario desea una apertura lineal mirando si la opción *"linear"* estaba presente en el fichero de configuración del *X-Window System*.
- Si estaba presente, verificaremos que el *chipset* es un *Accelerated'r'Us*, ya que el *Graphics'r'Us* no soporta apertura lineal. En ambos casos presentaremos las trazas adecuadas. Si el chipset es el primero, informaremos al servidor de que existe la capacidad de crear un *framebuffer* lineal.
- Rellenaremos el resto de los campos de la estructura del *chipset*, informando de la posición de inicio de la apertura, elegida en función del *BUS* y del tamaño de la misma.

Figura 3



Función GUInit():

- Al comienzo de cada modo gráfico manipularemos los registros que sean oportunos, de tal manera que cada una de las escrituras y lecturas de la tarjeta se realice en la apertura lineal que con anterioridad se indicó al servidor.

Ya hemos aprendido a comprobar si una opción ha sido indicada en **XF86Config**. Sin embargo, aún no sabemos la manera de informar al servidor de las capacidades de apertura lineal de nuestra tarjeta.

Cuando realizamos el *driver* para el *chipset Graphics'r'Us*, informamos de los tamaños de banco, número de bancos y otras características de la tarjeta en una estructura denominada GU, la cual era de tipo **vgaVideoChipRec**. Pues bien, las capacidades en cuanto a apertura lineal se refieren, también se han de reflejar en esta estructura GU, en los campos:

GU.ChipUseLinearAddressing
GU.ChipLinearBase
GU.ChipLinearSize

El código utilizado para informar al servidor de que la tarjeta tiene capacidad para soportar una apertura lineal en 0xE0000000, de un tamaño igual a la memoria detectada, se puede ver en el Listado 3.

Tabla 1

	8 bpp	16 bpp	24 bpp	32 bpp
Rojo	n/a	5 bits	8 bits	8 bits
Verde	n/a	6 bits	8 bits	8 bits
Azul	n/a	5 bits	8 bits	8 bits
Bytes Totales	1 byte	2 bytes	3 bytes	4 bytes

Las capacidades en cuanto a apertura lineal se refieren se reflejan en la estructura GU

Una vez hayan sido realizadas las modificaciones oportunas en **GUProbe()**, habrá que hacer lo propio en **GUInit()**. Para realizarlo, en función del *bus* que tenga la tarjeta, escribiremos los datos adecuados en el registro de apertura de memoria. Puerto 0x3DF, índice 0xA. Dependiendo del tipo de direccionamiento que se haya seleccionado, escribiremos en él los valores que indica el manual:

Modo Segmentado: 0x00

Apertura Lineal

ISA/VLB/MCA: 0x01
PCI (0xE0000000): 0x03

Una vez llevada a cabo la codificación, probaremos el servidor de la misma forma que hicimos en el artículo anterior. Desde el directorio raíz del *LinkKit* haremos:

```
./mkmf
make clean
make
```

Y probaremos el nuevo servidor con:
./XF86_SVGA.

En primer lugar en modo segmentado, para comprobar que funciona realmente y que no hemos estropeado nada, y en segundo lugar esta vez ya en modo lineal, incluyendo la línea que mostramos a continuación en el fichero:

/etc/XF86Config: Option "linear".

Antes de que nos pongamos a realizar la prueba conviene que ejecutemos un

par de *sync*. De esta manera, si el PC se queda *colgado*, lo cual es muy probable si no hemos tenido mucho cuidado, no perderemos ninguna modificación que hayamos hecho recientemente.

Más Allá: 16 y 32 bpp

Aunque parezca extraña la elección de implementar el direccionamiento lineal antes de añadir más colores, no nos quedaba otro remedio. La razón es muy sencilla: el servidor de X no permite modos de más de 256 colores en direccionamiento segmentado. Sin embargo, Añadir modos de más colores al servidor SVGA es fácil.

Los pasos que debemos seguir son los siguientes:

Función GUProbe():

- La primera acción que tenemos que realizar es comprobar que la tarjeta de vídeo de la que disponemos en el momento es una del tipo *Accelerated'r'Us*.
- Una vez sabemos que la tarjeta soporta modos superiores, interrogaremos al servidor para ver si la opción *"linear"* estaba presente en el fichero de configuración **/etc/XF86Config**.
- En caso afirmativo determinaremos, mediante accesos al registro de configuración, las capacidades del RAMDAC de la tarjeta. En función del resultado rellenaremos los campos relativos a la profundidad de color soportados por el *driver* en la estructura **GU**. Los campos relevantes son:

GU.ChipHas16bpp
GU.ChipHas24bpp
GU.ChipHas32bpp

- Mostraremos una traza por la salida estándar que informe de las capacidades del RAMDAC instalado.

Función GUInit():

- En función del valor de la variable global del servidor que indica la profundidad de los modos, configuraremos el registro del RAMDAC. La variable del servidor puede tomar cuatro valores: 8, 16, 24 y 32. Su nombre es: **vgaBitsPerPixel**

El servidor de X no permite modos de más de 256 colores en direccionamiento segmentado

Función GUAdjust():

- Esta función resulta de mucha utiliidad para colocar las direcciones de inicio de la parte visible de la pantalla virtual. Como estas direcciones se miden en píxeles, que están contados desde el origen de coordenadas, hemos de informar al servidor de cuántos *bytes* componen un píxel. En el caso de 256 colores esta información carecía de relevancia, ya que cada píxel ocupaba exactamente un *byte*. En la tabla 1 podemos encontrar la relación entre ambos para distintas profundidades.

Una vez más, solo falta compilar, enlazar y probar el *driver*. Para arrancar el servidor en un modo diferente a 256 colores se emplea la sintaxis:

./XF86_SVGA — -xxbpp

donde *xx* representa la profundidad deseada y puede tomar los tres valores siguientes: 16, 24 y 32.

Una vez hayamos comprobado que el código funciona perfectamente, podemos instalarlo en nuestro sistema mediante el comando: **make install** ejecutado desde el directorio raíz del *LinkKit*.

Conclusiones

En este segundo artículo de la serie dedicada a crear un *driver* para una nueva tarjeta de vídeo, hemos extendido la funcionalidad del código que escribimos en el número anterior de la revista, añadiendo soporte para:

- Múltiples *chipsets* en un mismo *driver*.
- Direccionamiento Lineal.
- Profundidades de color más allá de 256 colores.

Además, hemos aprendido a utilizar las opciones que se pueden pasar al servidor vía fichero de configuración. Finalmente, hemos recalcado la necesidad de utilizar trazas en el código del *driver*. En el próximo artículo continuaremos la serie, añadiendo soporte para el *Cursor Hardware*.

Por último, recordar que el código del *driver* se encuentra en el CD de la revista. De nuevo he de indicar a los lectores que los ficheros se proporcionan como guía ilustrativa y que, en ningún caso, se debe intentar añadirlos al servidor.

Contactar con el autor

Jorge Delgado Mendoza es miembro de XFree86 desde el año 1994, como desarrollador responsable del servidor de Oak Technologies Inc. Es Ingeniero Superior de Telecomunicaciones por la UPM y programador profesional. Puede ser contactado por correo electrónico en la dirección: **ernar@dit.upm.es**.



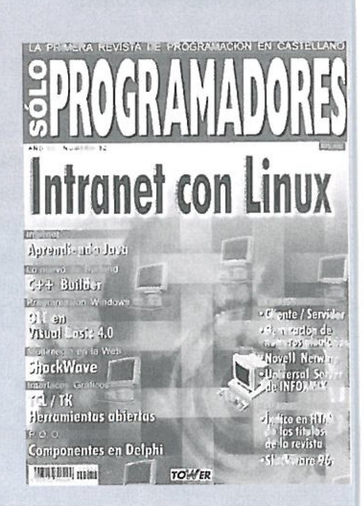
LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
BORLAND INTRABUILDER PROFESIONAL 1.0
COMUNICACIONES EN WINDOWS
PROGRAMACIÓN DE INTERFACES GRÁFICAS CON WINDOWS X
TOWER



LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
PROGRAMACIÓN EN JAVASCRIPT
GESTIÓN DE EVENTOS • SCRIPTS INCRUSTADOS COMPARATIVA JAVA, CGI
MULTITAREA EN TAPI
DEMOSCENE: MODELOS DE ILUMINACIÓN
BORLAND C++ 5.01
TOWER



LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
NIVEL DE USUARIO EN LINUX
DEMOSCENE: INCLUYENDO OBJETOS 3DS
SP
GESTIÓN DE PROTOCOLOS
NETWARE Novell
TOWER



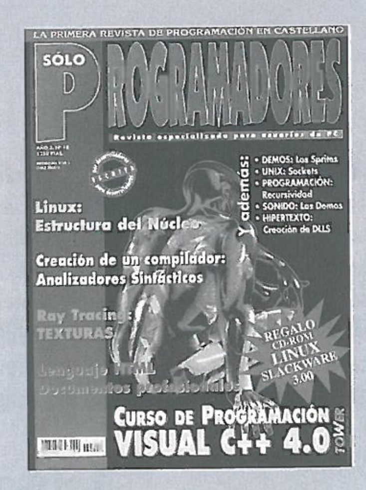
LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
Intranet con Linux
C++ Builder
Visual Basic 4.0
ShockWave
TCL / TK
Herramientas abiertas
Componentes en Delphi
Cliente / Servidor
TOWER

números atrasados

PROGRAMADORES

completa ya tu colección

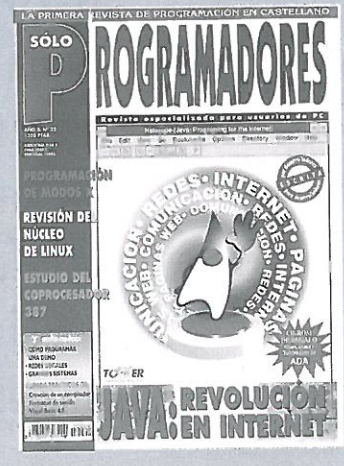




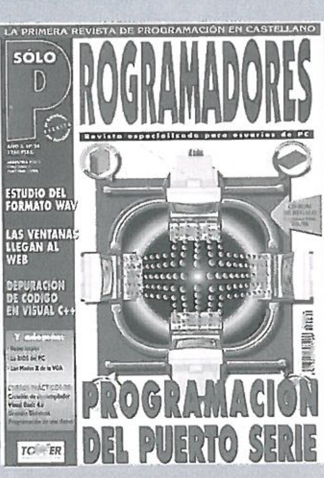




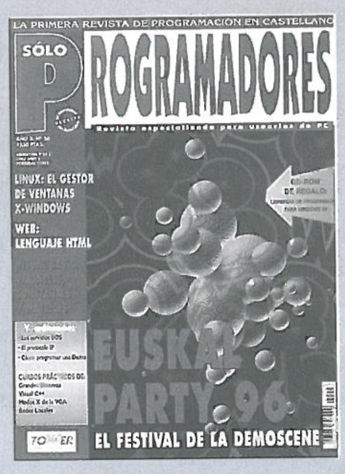




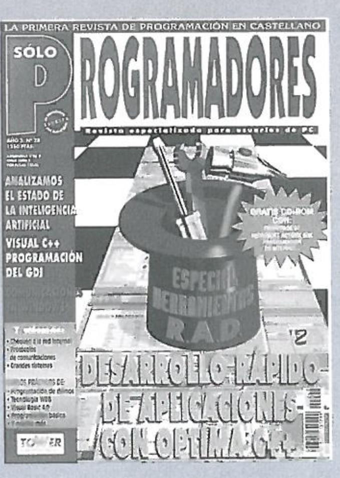












BASES DE DATOS

Oracle 8, abarcando las necesidades de la red

Carlos Urbano

Oracle 8 es el nuevo enfoque que desde la ya conocida Oracle dan a su servidor de base de datos. Entre sus principales funcionalidades se hallan el soporte para entornos Data Warehouse, el Soporte de Proceso de Transacción y la gestión de datos relacional-objeto.

Las nuevas tecnologías nos conducen a plantearnos horizontes renovados en nuestros sistemas de tratamiento de la información que se encuentran en evolución hacia arquitecturas abiertas. Estas tecnologías tienen como fundamento la necesidad de ampliar las formas de acceso a los datos, pudiendo encontrarse éstos en diferentes lugares, ya no dentro de nuestra empresa, sino en sitios separados miles de kilómetros.

Las redes y la informática distribuida son parte integrante de la nueva concepción de las arquitecturas de los sistemas de información. Cuando desde una organización surge la necesidad de adaptarse a esta concepción del tratamiento de la información, los responsables de decidir la adquisición de la mejor opción entre la maraña de productos existentes en el mercado se enfrentan a diversos retos:

- Mantener el rendimiento y la robustez de los sistemas existentes en los sistemas nuevos que se encuentran en fase de desarrollo.

- Identificar los mejores productos que se pueden implantar en ese entorno.

- Conseguir la interiperatividad entre los datos nuevos y los ya existentes.

- Integrar y gestionar un entorno multinivel y multiplataforma.

Oracle ha sacado al mercado recientemente la nueva versión del servidor de base de datos Oracle 8, como adaptación de Oracle 7 a las nuevas necesidades que genera el llamado *Network Computing* y como posible solución para los retos definidos anteriormente, en medianas y pequeñas empresas.

Network Computing es un término que hace referencia al ordenador orientado a la informática de la red, Internet e intranets

Como nuevas características del producto podemos citar su alta disponibilidad y capacidad de gestión con tablas e índices divididos en particiones, el soporte de un mayor número de usuarios y volúmenes de información, mayor rendimiento, paralelismo mejorado, mejoras en la gestión del aplicaciones de DataWareghouse y OLTP, en niveles de optimización parecidos a los obtenidos por los *mainframes,* mejor administración de la seguridad, informática distribuida avanzada, tecnología de objetos y capacidad de ampliación, he-

rramientas de gestión de fácil utilización y una migración sin problemas desde versiones anteriores.

Características del sistema

La arquitectura de software para la Informática en Red de Oracle es abierta, está basada en estándares y pretende resolver los problemas de interoperatividad entre diferentes entornos. Su componente central, el *Oracle Server,* es el responsable directo del incremento del rendimiento del producto, permitiendo la ejecución de aplicaciones de gran tamaño como son las *Data Warehouse*.

Entornos Data Warehouse

En entornos *Data Warehouse,* Oracle 8 soporta funciones de procesamiento mejorado de consultas en estrella y soporte para mayor número de operaciones en paralelo. A partir de cualquier tipo de fuente de datos (Informix, DB2, Sybase, etc.) permite crear un DW y gestionar cualquier tipo de dato: texto, vídeo, sonido, etc.

Los problemas en los entornos DW están tradicionalmente ligados a la gran cantidad de información que tienen que ser capaces de gestionar, y Oracle 8 ha sido concebido para solucionarlos:

• Soporte de petabytes de datos: los Dw soportan un volumen creciente de datos, en la actualidad del orden de gigabytes de información, Oracle 8 es capaz de gestionar petabytes de datos (mil terabytes) al mismo tiempo que permite almacenar hasta mil columnas en cada tabla correspondientes a los atributos de un dato. Normalmente, manejar este volumen de información repercute en un decremento del rendimiento, para evitar esto se han introducido funciones avanzadas de particionamiento de datos, es decir se subdividen las tablas de datos en particiones múltiples, y se ha introducido la paralelización, permitiéndose ejecutar en paralelo distintos procesos de acceso a las bases de datos, como son inserciones, consultas, borrados, etc.

El particionamiento de los datos se realiza de forma inteligente, es decir, el servidor de Oracle 8 divide las tablas de datos en diferentes segmentos según las necesidades de cada usuario, de forma que, cuando se realiza una consulta, el servidor analizará únicamente las particiones de información que hayan sido señaladas y no la totalidad de los datos, con lo cual se obtiene como resultado un incremento del rendimiento y un ahorro sustancial de tiempo.

A partir de cualquier tipo de fuente de datos (Informix, DB2, Sybase, etc.) permite crear un DW y gestionar cualquier tipo de dato

Estas características son independientes de la plataforma hardware y la arquitectura, ya que el producto soporta varios entornos de gestión de datos distintos: Ncs, Unix, Windows NT, Pcs, Sistemas Masivamente paralelos, Sistemas de Prooceso Paralelo, entornos Cluster y servidores centrales o mainframes.

• Mayor número de usuarios: el número de usuarios que acceden a un sistema de Data Warehouse ha ido en ascenso a medida que se ha comprobado la utilidad de los entornos DW; este incremento suele implicar graves problemas de rendimiento debido a la sobrecarga producida.

Las consultas en estrella se utilizan de forma habitual en las aplicaciones DW cuando se establecen relaciones entre una o más tablas muy grandes con otras de tamaño más reducido

Además, no todas las bases de datos son capaces de soportar miles de usuarios, lo cual puede ser una cifra que se maneja de forma habitual dentro de una gran organización. A través de navegadores web y de aplicaciones cliente/servidor, Oracle 8 proporciona acceso a la información del DW sin reducir el rendimiento global del sistema.

• Rapidez de acceso: desde los comienzos de la utilización de la informática en el mundo empresarial la velocidad de proceso de los datos ha sido un punto crítico en los sistemas de información. Es muy importante que el acceso a los datos sea muy rápido; Oracle 8 proporciona una serie de herramientas de acceso a la información que posibilita maximizar esta necesidad.

• Disponibilidad asegurada: debido a la importancia creciente que los DW están adquiriendo en el mundo em-

presarial, se tiene que poder disponer de ellos en cualquier momento, 24 horas al día, todos los días del año.

Para evitar caídas del sistema Oracle 8 dispone de un sistema de seguridad que recupera de forma automática los datos.

Las consultas en estrella se utilizan de forma habitual en las aplicaciones DW cuando se establecen relaciones entre una o más tablas muy grandes con otras de tamaño más reducido. Oracle 8 proporciona una forma de ejecución de este tipo de consultas que incrementa el rendimiento de las aplicaciones DW.

En entornos DW, así como en entornos de soporte para la toma de decisiones, se utilizan con frecuencia las operaciones en paralelo para conseguir una mayor rapidez en todos los procesos. Con la nueva versión de Oracle se ha permitido la realización de operaciones como la inserción, la actualización y la eliminación de los datos en paralelo, de forma que su ejecución se realiza mucho más de prisa que si ésta fuera en serie. Esta característica también resulta de utilidad en las bases de datos de OLTP, para optimizar los trabajos por lotes.

Procesamiento OLTP

Oracle 8 está diseñada para soportar hasta centenares de miles de usuarios de forma nativa y bases de datos de tamaño muy elevado, del orden, como mencionamos anteriormente, de los petabytes de datos. Esta característica hace de esta base de datos un producto apropiado para ser utilizado en el soporte de procesos de transacciones on-line (OLTP), siendo, como se indica desde Oracle, la primera base de datos que cumple los requerimientos técnicos necesarios para trabajar con sistemas OLTP. Esta característica tiene importantes implicaciones debido sobre todo a que la mayoría de las bases de datos actuales se utilizan en la gestión de aplicaciones de este tipo (junto con las DW). Teniendo en cuenta este factor, la nueva versión del producto ha intentado mejorar su diseño, y estas mejoras han sido orientadas, sobre todo, a los puntos siguientes:

- Acceso mejorado a la base de datos. Con esta versión se puede soportar un entorno con más de 4000 usuarios con sistemas abiertos o con mainframes: ya no es imprescindible utilizar software o hardware de multiplexación (lo que se conoce por monitores de transacciones) junto con un mainframe para obtener un tiempo de respuesta óptimo cuando se conecta un número de usuarios elevado.

Ya no es imprescindible utilizar monitores de transacciones junto con un mainframe para obtener un tiempo de respuesta óptimo cuando hay un número de usuarios elevado

- Recuperación automática de datos. El *Oracle Parallel Server* se encarga de realizar la recuperación automática en una eventual caída del sistema. Cuando un nodo del *Parallel Server* tiene un fallo, una función realiza la migración de las conexiones de los usuarios y restablece sus sesiones en otro nodo de forma transparente. De esta manera las aplicaciones pueden continuar ejecutándose. Este proceso se puede realizar además cuando existe un número de usuarios conectados excesivo, trasladando sus sesiones a otro nodo del servidor.

- Desconexión automática de usuarios inactivos. Esta mejora del servidor de Oracle 8 está pensada para aumentar la utilización de los recursos del sistema operativo y de la red. Cuando un usuario no está utilizando su conexión, ésta se desconecta de forma temporal y se vuelve a restablecer sin que el usuario lo perciba, incrementando de esta manera el número de usuarios.

- Operaciones avanzadas de colas. Este tipo de operaciones permiten retrasar la ejecución de transacciones y ejecutarlas en un orden determinado, de esta forma se elimina la dependencia de sistemas externos para las aplicaciones que necesitan un alto grado de escalabilidad. Esta función se utiliza para realizar la implantación de aplicaciones de flujo de trabajo que trasladan datos a un sistema a medida que cambia el estado de los mismos.

- Copias de seguridad y recuperación de los datos. Oracle 8 mantiene información sobre el momento en que se realizan las copias de seguridad y sobre el lugar donde se encuentran almacenados los archivos; si fuera necesario recuperar los datos, el propio servidor se encarga de definir el proceso, determinando el estado de la base de datos y la forma de actuación para su reparación. Estas operaciones tienen lugar de forma automática, sin intervención de ningún operario, lo cual reduce las posibilidades de que se produzca un potencial fallo humano.

- Partición inteligente de los datos. Oracle 8 permite realizar la

división de las tablas y los índices en particiones, o partes más pequeñas, para mejorar de esta forma el rendimiento y la administración de los sistemas basados en DW y OLTP: los datos que se encuentran en una tabla que se divide de esta forma están disponibles aunque una o más particiones no lo estén , ya que las particiones funcionan de forma independiente. Con esta división de las tablas, se consigue facilitar la gestión de las tablas de gran tamaño ya que las operaciones que se realicen sobre ellas pueden ser subdivididas en otras de menor tamaño que pueden, como ya explicamos, realizarse de forma paralela.

Existe además un optimizador en Oracle 8 que es consciente de la existencia de las particiones y decide cuáles son necesarias en el proceso de realización de una búsqueda, eliminando del proceso las que no lo sean porque no contengan ningún dato que participe en la búsqueda, aumentando de esta forma el rendimiento.

Oracle 8 permite dividir las tablas y los índices en particiones para mejorar el rendimiento y la administración de los sistemas basados en DW y OLTP

• Seguridad incrementada. La seguridad es un aspecto muy importante dentro de los sitemas de gestión de bases de datos, los datos no pueden estar a disposición de personas que no deben leerlos, por razones que todos podemos entender. La nueva versión de Oracle incluye un servidor de seguridad que establece un entorno único de administración central de usuarios y funciones: *Oracle Security Server.* Admite la autenticación entre cliente y servidor para evitar la captura ilegal de datos entre sistemas clientes y ofrece una serie de herramientas que generan firmas digitales para permitir la creación de aplicaciones para identificar la manipulación no autorizada de los datos.

El servidor cumple la norma de seguridad X.509 para la autenticación de las claves públicas y privadas.

Base de datos relacional orientada a objetos

Los nuevos tipos de datos que la empresa genera y requiere gestionar repercuten en la necesidad de definir un modelo actualizado para poder realizar la gestión de las bases de datos. Ante la obligación de simplificar la modelización de los datos y de ampliar la BD con los nuevos tipos, el sistema de gestión de base de datos Oracle se ha adherido a la tendencia de utilizar la tecnología relacional-objeto dentro de la implementación de los sistemas de gestión.

Oracle 8 permite definir tipos de objetos para contemplar la posibilidad de manejar tipos no estructurados complejos

La complejidad de las bases de datos y de las aplicaciones que las utilizan aumenta, lógicamente, a medida que lo hacen los tipos de datos que deben ser capaces de gestionar. Además, es habitual que varias aplicaciones que son independientes entre sí tengan datos similares y que estos datos se encuentren almacenados en distintas partes de la base de datos sin relación aparente entre ellas. Esta característica propia de las BD provoca también que se complique la gestión de los datos mucho más de lo que sería conveniente.

La información debe ser tratada de forma totalmente coherente si se desea conseguir que los distintos elementos que forman parte de un sistema de gestión de base de datos se integren de manera adecuada. La integración que se consigue con la nueva versión del servidor Oracle se obtiene, a grandes rasgos, con la utilización de los denominados objetos relacionales, que se acoplan a las distintas aplicaciones que conforman un sistema de información con el fin de dar forma a un modelo de datos que sea uniforme. La utilización de los objetos relacionales se engloba dentro de la tecnología relacional orientada a objetos que ya otras empresas del sector utilizan en sus sistemas de gestión de bases de datos, como es el caso de Informix.

Las características relacionales orientadas a objetos de Oracle 8 incluyen el soporte de datos no estructurados, la adopción de las normas estándar del sector para Java, la definición de nuevos tipos de objetos para ampliar el servidor de forma específica en cada aplicación y otros, que explicamos un poco más detalladamente a continuación:

• Soporte de los datos no estructurados. Las bases de datos relacionales tradicionales admiten, por lo común, manejar tres tipos de datos: caracteres, números y fechas; en la versión Oracle 8 se permite definir distintos tipos de objetos para poder contemplar la posibilidad de manejar tipos no estructurados complejos, como datos multimedia, imágenes, vídeo o texto.

Mediante esta interesante característica tenemos la posibilidad de tratar diferentes áreas de negocio, como puede ser la edición profesional de todo tipo de libros y películas, o trabajos para la prensa y la televisión. También es importante tener en cuenta las nuevas empresas que utilizan de alguna manera elementos multimedia, abarcando diversas especialidades: desde la creación de software que esté orientado al ocio hasta las diversas necesidades que supone la utilización masiva de la tecnología Internet, que también ha generado nuevas necesidades en cuanto a los tipos de datos que trata y genera y la forma que tiene de procesarlos, etc.

• Soporte para Java. Para conseguir el soporte de Java de forma completa, Oracle se encuentra participando en el desarrollo de tres iniciativas, alguna de ellas ya disponible: un driver JDBC suministrado por Oracle que se integra con los tipos de objetos de Oracle; J/SQL para incrustar intrucciones SQL en el código Java; y una máquina virtual JAVA que se encuentra situada en la base de datos y que ha sido concebida para almacenar y ejecutar el código Java dentro del motor de la base de datos de Oracle.

La primera opción, JDBC, está ya disponible en el mercado para clientes Java para permitir el acceso a Oracle 8.

El JDBC para Java suministrado por Oracle se integra con los tipos de objetos de la base de datos

• Herramientas de desarrollo. Varias herramientas de desarrollo como Designer/2000 y Sedona, admiten en la actualidad el modelo de objetos de Oracle 8 y se prevee que en breve el abanico de herramientas se emplíe.

Desde Oracle nos promocionan este producto como una solución que aporta la potencia, el rendimiento, la integración en red y la felxibilidad necesarias para convertirse en el sistema de gestión de bases de datos del momento. Su utilización dentro de la empresa y en las aplicaciones de negocio nos dirán si realmente es lo que parece.

Programación de emuladores

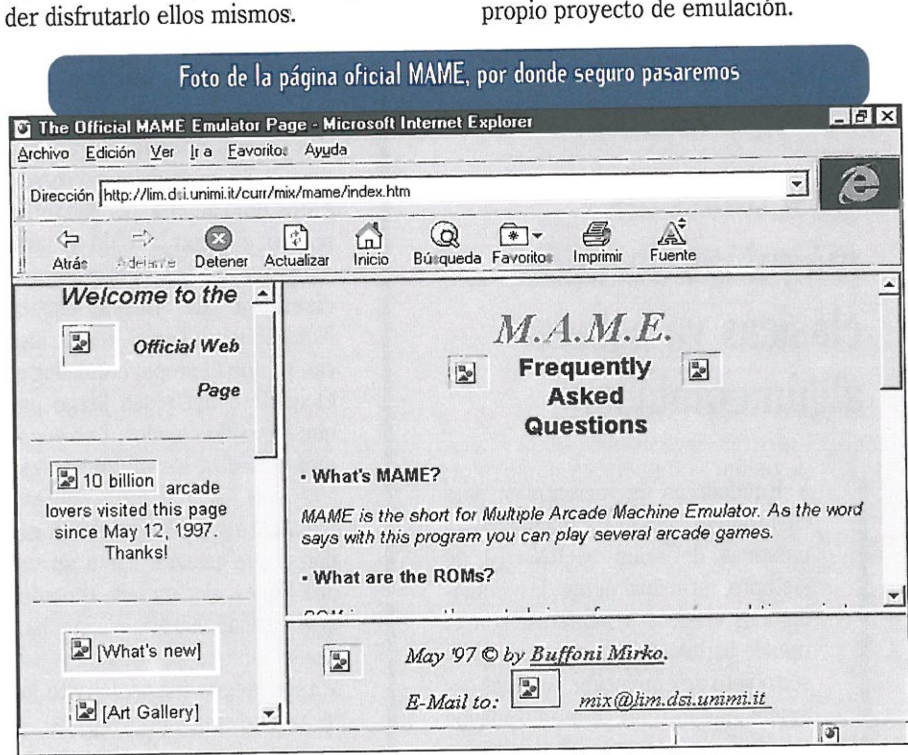
Jorge Figueroa

Desde hace algún tiempo, la programación de emuladores de toda clase de vídeo consolas y ordenadores en general, se ha convertido en un hobby muy extendido tanto entre aficionados como entre expertos de la programación, que se dedican, en sus ratos libres, a crear emuladores de aquello que siempre habían querido tener pero que no estaba disponible en su plataforma actual (diferente arquitectura o simplemente obsoleta y olvidada) con el objetivo de ganar algo más de dinero extra (emuladores shareware), para saber si son capaces de realizarlo o, simplemente, para poder disfrutarlo ellos mismos.

Para el programador clásico que nunca ha tocado este tema y que quiera iniciarse en él, quizás al principio le parezca todo muy secreto y desconocido. Por ello, para aquellos que no saben cómo empezar y para animar a los que aún no le han hecho caso al tema, se van a explicar todos los conceptos básicos relacionados con la programación de emuladores, incluyendo las definiciones, las técnicas de programación más utilizadas y algunos consejos generales que servirán como ayuda para que cualquier programador pueda empezar con buen pie su propio proyecto de emulación.

La programación de los llamados emuladores es una de las áreas que actualmente se está haciendo muy popular entre los programadores tanto profesionales como aficionados de todo el mundo. En este artículo presentamos toda la información técnica necesaria para aquellos que quieran adentrarse en este apasionante hobby.

Foto de la página oficial MAME, por donde seguro pasaremos



La programación de emuladores es un hobby en auge

Antes de comenzar, sólo mencionar que si quieren mandar algún mensaje sobre éste o cualquier otro artículo del autor, pueden entrar en la WEB para programadores <"www.arrakis.es/~erde/index.htm> y dejar un mensaje al autor.

Estado actual de la emulación

Actualmente el interés por el tema, como hemos dicho, se ha extendido mucho y podemos encontrar emuladores de casi cualquier cosa que posea un microprocesador y circuitos.

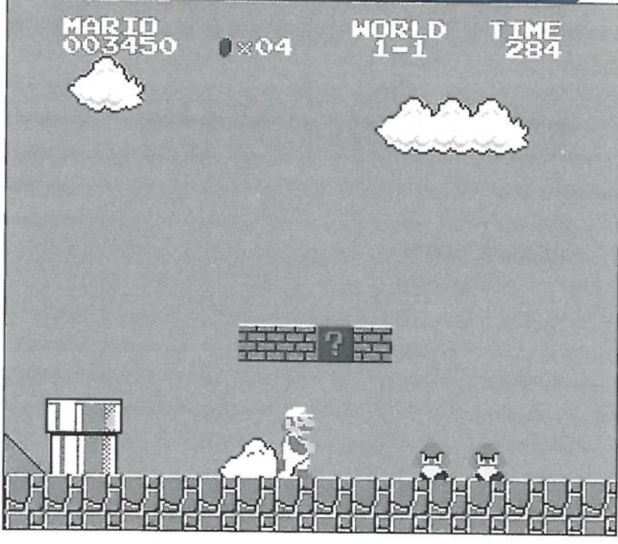
En términos generales podemos decir que existen los siguientes tipos de emulador, que corresponden más o menos a las diversas clases de plataformas existentes según su área de uso:

- Emuladores de calculadoras: pese a que quizás es el área menos conocida, existen también programadores que realizan emuladores de calculadoras de todas las clases, normalmente científicas y similares y que pueden resultar un buen entretenimiento.

- Emuladores de ordenador: como todos sabemos, en los últimos quince años han aparecido una gran diversidad de ordenadores domésticos destinados sobre todo para su uso como plataformas de juego (entretenimiento) de bajo coste. Podemos decir que prácticamente no queda ninguna máquina de este tipo sin emular y los programas disponibles poseen en algunos casos una calidad excepcional (o profesional mejor dicho). Así, por ejemplo, disponemos de emuladores de Spectrum, MSX, Atari, Comodore Amiga, Amstrad CPC, etc.

- Emuladores de videoconsola: el área de las vídeo consolas es una de las más explotadas y existen para cada una de las máquinas conocidas del mercado más de un emulador de calidad, excepto, claro está, para las de última generación, que, por falta de tiempo y/o medios, no han podido ser todavía emuladas. Sólo resaltar que por medios, nos referimos a que la máquina donde tiene que realizarse la emulación tiene una potencia de procesamiento (incluyendo la gráfica y sonora) muy inferior a la máquina a emular y que, por lo tanto, el proyecto es técnicamente inviable, ya que no podría conseguirse un funcionamiento mínimamente similar al de la máquina original, característica sin la cual el programa carecerá de interés y utilidad, ya que no podremos ejecutar su soft.

Poder jugar con clásicos como Mario Bros ha dado popularidad a los emuladores.



Casi todas las máquinas de salón clásicas ya poseen algún emulador

- Emuladores de recreativas: para casi todo el mundo las máquinas recreativas de salón (COIN-OP) de siempre, principalmente las consideradas clásicos y pioneras, son el tipo de hardware que más atrae para ser objeto de emulación. Además tenemos que el tema posee un mayor atractivo para los posibles usuarios que van a disfrutar del programa, lo que da a su vez un incentivo añadido para el programador que sabe que su creación será más usada que si realiza la emulación de, por ejemplo, un MSX o una Lynx.

Dentro del apartado de emuladores de recreativas, nos encontramos que tenemos dos familias principales de programas. Por un lado tenemos los llamados *single emulators* y por otro los llamados *multiple-game emulators*. Los *single emulators*, como su nombre indica, son aquellos que se han diseñado para emular a un sólo juego de máquina recreativa. Estos emuladores, por lo tanto, sólo pueden ejecutar la ROM de un único vídeo juego para el que hayan sido diseñados de forma específica. Normalmente son fáciles de identificar, ya que el propio emulador posee el nombre del vídeo juego para el que han sido creados. Los segundos mencionados, los *multigame emulators*, son los más interesantes normalmente y corresponden a emuladores que pueden ser a su vez de dos clases. Por un lado tenemos los que emulan hardware que ha sido usado como plataforma base para diversos juegos (normalmente juegos de una misma casa y época), como es, por ejemplo, la que usó SEGA

como plataforma de sus primeros arcades de salón, y en cuyo caso tendremos entonces que todas las ROMS (juegos) que fueron diseñados para dicha plataforma, podrán funcionar en ese emulador sin problemas. Por otro lado, tenemos lo que corresponde a la segunda subclase de emuladores *multigame* que se caracterizan por que soportan en un sólo programa diversas plataformas y que ejecutan unas u otras partes del código según la ROM que queramos ejecutar. Uno de los *multigame emulator* de este tipo que se está convirtiendo en la estrella mundial, es el llamado MAME, el cual es un proyecto en el que está participando mucha gente y con el que, a partir de un sólo emulador, se pueden jugar a un número cada vez mayor de vídeo juegos (más de 120). Su éxito es tal, que se está portando a muchas plataformas (o sea, un emulador multiplataforma disponible en varias plataformas).

• Otros emuladores: a parte de los ya mencionados, también hay emuladores más clásicos, como eran los ya famosos del coprocesador matemático 387 y otros dispositivos como tarjetas de sonido o impresoras, incluyendo en este apartado casi cualquier cosa que posea microprocesador.

Legalidad de los emuladores

La legalidad de emular máquinas ha sido un tema muy discutido durante mucho tiempo pero en resumen podemos decir que sólo es ilegal distribuir ROMs que estén protegidos por copyright (y sólo en algunos países), como son las ROM BIOS que pueda tener un ordenador, o los vídeo juegos de arcades y vídeo consolas.

Así pues, si vamos a distribuir algún día un emulador creado por nosotros y nos queremos asegurar de que no tendremos ningún problema, deberemos recordar que nunca debe incluirse ROM alguna junto con él ya que posiblemente estaremos infringiendo la ley.

Técnicas generales de programación

Hay básicamente tres técnicas de programación de emuladores distintas: podemos encontrarnos con los llamados interpretes, los de recompilación estática y los de recompilación dinámina.

• Emulador Intérprete

Un emulador intérprete es aquel que emula el código de las ROMs que ejecuta leyendo byte por byte los fuentes originales de la misma y que realiza lo necesario para emular todo el funcionamiento lógico de cada una de las líneas de código que encuentra.

Esta clase de emulación es la más fácil de programar y de depurar, pero a su vez, los ROMs (programas) que se ejecuten con este sistema serán mucho más lentos, dado que se realizan todos los cálculos en tiempo real.

• Recompilación estática

La recompilación estática consiste en que las ROMs a ejecutar se leen primero por completo y se convierten los fuentes de la misma a un código ensamblador equivalente de nuestra máquina.

Este sistema de ejecución es el mejor cuando la ROM cumple algunos requisitos, ya que la velocidad de ejecución es la máxima que se puede obtener, aunque si se da el caso de que el programa original poseyera código automodificable, sería imposible convertir el código desde un ensamblador a otro, por lo que este sistema quedará descartado.

Hay dos técnicas principales de emulación: interpretación y recompilación

• Recompilación dinámica:

Este método es básicamente idéntico al anterior en cuanto a la forma de ejecutar las instrucciones de la ROM pero con la diferencia de que la recompilación se realiza sobre la marcha en bloques de código según se va ejecutando la ROM en vez de realizar la recompilación en un sólo paso.

Como puntos de referencia el emulador suele detectar la aparición de instrucciones de salto para saber cuándo se cambia el bloque de ROM que se requiere ejecutar para proceder a la recompilación del nuevo bloque requerido (una especie de sistema de paginación de código).

Este sistema es muy usado en emuladores de ordenadores, como puede ser el de Macintosh y, aunque da un rendimiento algo inferior al método estático, es más flexible.

El código automodificable siempre es lo más difícil de emular

Otra posibilidad que se empieza a usar últimamente es la de convertir los ASM originales a código intermedio, con

lo cual puede mejorarse la velocidad de ejecución. Esta técnica la podríamos comparar al sistema usado en el JAVA, donde los fuentes se convierten en un pseudocódigo (byte-code) que después se ejecuta en cualquier plataforma, lo que no puede considerarse ni interpretación ni compilación pura.

Comenzando el emulador

Cuando hayamos decidido qué plataforma hardware y ROMs (en caso de que sea una recreativa) queremos emular, deberemos empezar por elegir el lenguaje de programación que utilizaremos. Como era de esperar, si examinamos las posibilidades, veremos que sólo tenemos dos opciones claras, que son el ensamblador y el lenguaje C. Más o menos todos conocemos los pros y los contras de cada uno de ellos y supongo que casi todo el mundo preferirá el lenguaje C, ya es el que ofrece más ventajas en todos los sentidos. Es un estándar mundial, está adaptado a casi todas las arquitecturas (plataformas), es el más rápido después del ensamblador y es muy fácil de depurar y actualizar posteriormente. Además de lo dicho, si nos encontramos que realizando el emulador totalmente en C el resultado no ofrece una velocidad mínima de ejecución, podremos reescribir las funciones más ejecutadas (críticas) en ensamblador, gracias a lo fácil que resulta también usar ensamblador en C, pudiendo incluso, en algunos casos, meter código *Asm inline* dentro de los propios fuentes C.

Procesadores que emular

Cuando tengamos toda la información técnica necesaria para saber cómo funciona el hardware que emular, deberemos antes que nada crear emuladores de cada uno de los chips que se encuentren instalados en el hard de la plataforma original. Así pues, por ejemplo, si queremos emular la máquina recreativa que usaba un microprocesador Z80 para ejecutar el código de programa y un chip de sonido para los efectos, deberemos crear un *engine* de ejecución para el set de instrucciones de cada uno de dichos chips.

Si queremos emular chips como el Z80, disponemos de engines ya creados

Si nuestro proyecto no es muy ambicioso ni muy vanguardista, seguramente los chips a emular serán muy conocidos y podremos conseguir en alguna WEB *engines* de emulación ya creados para dichos microprocesadores. Un ejemplo de CPU muy conocida y usada es el ya mencionado Z80, el 6502 y el 6809. Algunos de estos *engines* de emulación están escritos en lenguaje C (son portables a cualquier plataforma) y han sido ampliamente usados en diversos proyectos conocidos, con lo cual están muy testeados y garantizados, lo que nos ahorrará infinidad de horas de programación y pruebas.

A continuación damos un enlance (*link*) donde podemos obtener algún que otro *engine* de emulación e inormación variada que puede sernos de interés: en la página www.lim.dsi.unimi.it/curr/mix/mame/index.html tenemos, bajo el título de "*The New Emulation Programming Repository*" varias versiones de MAME, utilidades para el mismo, emuladores y *engines* de los mencionados, como son el conocido emulador de Z80 de Marat Fayzullin (que ha sido utilizado en más de diez proyectos terminados, como es el propio MAME, entre otros), fuentes de un emulador del chip 6502, dos emuladores del MC 68K, algunas rutinas para manejar el POKEY. (sistema de sonido de algunas antiguas recreativas), entre otras informaciones también muy interesantes.

Creando un engine

Si debemos emular algún micro del cual no disponemos de ningún *engine* ya creado o simplemente preferimos crearnos el nuestro propio por la razón que sea, lo primero que deberemos encontrar es un buen libro de programación ensamblador de dicho micro ya que, para poder emularlo, lógicamente necesitamos tener información a mano de absolutamente todo su funcionamiento interno, incluyendo su set de instrucciones completo, los registros que usa, cómo se codifican las instrucciones a nivel binario, tiempos de ejecución de las mismas, interrupciones que se producen y otros datos importantes que pueda haber. Por poner un caso y para seguir los ejemplos, si deseáramos toda la información necesaria del chip Z80, con el libro "Programación del Z80", del autor Rodnay Zaks (editorial Anaya Multimedia), ya tendríamos toda la información necesaria para poder realizar el proyecto.

Información de la plataforma

Una vez tenemos los emuladores de los chips, deberemos obtener toda la información técnica que podamos de la arquitectura física de la plataforma a copiar (en cuanto a funcionamiento lógico se refiere, claro está). Por ejemplo, si se trata de una maquina de salón, deberemos intentar conseguir lo que llamamos mapas de memoria.

En el listado 1 hemos puesto el ejemplo de un pequeño mapa de memoria del juego MISSILE COMMAND, una máquina de salón que apareció hace muchos años.

De estos mapas, como se puede observar en el ejemplo, podremos obtener toda la información necesaria de las áreas de memoria (como es el caso), puertos de comunicación de todos los dispositivos, funciones programables, etc., que tenga

el sistema, con lo cual ya podremos empezar la programación del emulador de la máquina, de la plataforma.

Creando el esqueleto

Cuando hayamos decidido el lenguaje que queremos usar, el hardware y soft que queremos emular y tengamos toda la información técnica y *engines* necesarios para empezar (lo cual ya es bastante) empieza lo que es la programación del emulador en sí.

Actualmente hay en desarrollo incluso emuladores de NEOGEO

Normalmente, el esqueleto general de un emulador es bastante parecido en todos los casos y su contenido podría corresponder al siguiente esquema funcional:

1- Ejecutar instrucciones del microprocesador(es) principal(es).
2- Detectar si se produce una interrupción Hblank por finalización de un *scanline*.
3- Detectar si se produce una interrupción Vblank por finalización de pantalla.
4- Gestionar los *sprites* (funcionalidades que dependen del hard).
5- Actualizar contadores de los temporizadores de la plataforma original y generar interrupciones en caso requerido.
6- Actualizar pantalla (total o parcialmente).
7- Gestionar el sonido (tanto FM como Digital).
8- Control de los dispositivos de entrada (teclado/ Joystick/ Trackball).
9- Control de funciones propias extras que se hayan incluido.

Listado 1. Ejemplo de un mapa esquemático del hard de la máquina recreativa MISSILE COMMAND.

```
HEX    R/W  D7 D6 D5 D4 D3 D2 D2 D0 función
---------+----+---------------------+-------------
0000-01FF R/W  D D D D D D D D 512 bytes RAM libres
0200-05FF R/W  D D D D D D D D Bit color 3º región de
la memoria de pantalla
0600-063F R/W  D D D D D D D D  Ram disponible
06F0-3FFF R/W  D D D D D D D D Bit color 2º región
de la memoria de pantalla
---------+----+---------------------+-------------
4000-400F R/W  D D D D D D D D  puertos POKEY.
4800   R  D                COIN Derecho
4800   R   D               COIN Central
4800   R    D              COIN Izquierdo
4800   R     D             1 jugador start
4800   R      D            2 jugador start
4800   R       D           2nd jugador disparo izquierdo
4800   R        D          2nd jugador disparo central
4800   R         D         2nd jugador disparo derecho
---------+----+---------------------+-------------
4800   R         D D D D  Horiz trackball displacement
Si ctrld=Alto.
4800   R  D D D D         Vert trackball displacement
if ctrld=Alto.
---------+----+---------------------+-------------
4800   W  D                No usado?
4800   W   D               Flip de pantalla
4800   W    D              Contador COIN izquierdo
4800   W     D             Contador COIN central
4800   W      D            Contador COIN derecho
4800   W       D           LED inicio 2º juegador.
4800   W        D          LED inicio 1º jugador
4800   W         D         CTRLD, 0=leer switches,
1= Leer trackball.
---------+----+---------------------+-------------
4900   R  D                VBLANK read
4900   R   D               Autotesteo del switch de entrada
4900   R    D              SLAM switch de entrada
4900   R     D             Horiz trackball dirección entrada
4900   R      D            Vert trackball dirección entrada
4900   R       D           1º jugador disparo izquierdo
4900   R        D          1º jugador disparo central
4900   R         D         1º jugador disparo derecho
---------+----+---------------------+-------------
4A00   R  D D D D D D D D  Option switches
4B00-4B07 W        D D D D  Color RAM
4C00   W                   Watchdog
4D00   W                   Reconocimiento Interruptción
---------+----+---------------------+-------------
5000-7FFF R  D D D D D D D D  Programa
---------+----+---------------------+-------------
```

Listado 2. Listas de Links importantes para el emulador MAME y otros.

WEB's oficiales MAME:
WEB oficial MAME PC (original):
<http://lim.dsi.unimi.it/curr/mix/mame/index.htm>.
WEB oficial MAME para Macintosh:
<http://www.primenet.com/~bradman/mame/>.
WEB oficial MAME UNIX:
<http://www.dit.upm.es/~jantonio/mame/>.
WEB oficial MAME Amiga:
<http://www.students.uiuc.edu/~bw-evans/amiga.html>.

Otras interesantes
Kosmos Software:
<http://www.arrakis.es/~erde/index.htm"
Atmospherical Heights:
<http://www.xs4all.nl/~delite/arcade_mame.html>
Dave's Video Games Classics:
<http://www.gamepen.com/gamewire/classic/classic.html>
Australian MAME Distribution Site:
<http://www.vds.com.au/~kingey/mame/main.htm>
MAME Fan Page:
<http://www.isd.net/jturgeon/mame/>
Lord 13's MAME Page:
<http://www.wwnet.com/~lord13/mame/mame1.html>
Cubeman's Arcade Game Highscore List:
<http://web.idirect.com/~cubeman>
Classic Arcade Emulators High Scores:
<http://www.primenet.com/~dmarli/arcade.html>

Links especiales con Librerías de ROM's recreativas:
Brian Peek's archivo de ROM's Arcade:
<http://www.vu.union.edu/~peekb/arcade/index.html>
Tant archivo ROM Arcade:
<ftp://ftp.tant.com>
Archivo AROM:
<http://services.worldnet.net/winter/arom.htm>

Trucos de programación

Como suele ser habitual en muchas de las facetas de la programación, siempre que se puede los programadores usan trucos para poder emular la plataforma original de forma más rápida.

Un ejemplo muy claro de ello lo tenemos en uno de los dos emuladores que existen de la máquina recreativa RYGAR, concretamente el del autor (principal) Jean-Marc Leang. En este caso, teníamos que la máquina original generaba sonido FM en tiempo real por medio de dos microprocesadores Z80. Como es lógico, tener que emular a dichas CPUs resultaría muy costoso en cuanto a recursos y sus autores tuvieron la genial idea de calcular (generar) todo el sonido FM de dichos chips de una sola vez, guardar en disco el sonido en forma de ficheros y posteriormente usarlos como si fuesen sonido digital grabado típico durante el juego al estilo de como se usan tablas matemáticas trigonométricas precalculadas en los *engines* de los juegos 3D actuales (por establecer una comparación).

Otra característica muy común que suelen incluir los programadores en muchos emuladores es la utilización de una opción de *Salto de Frames* que consiste en que sólo se actualizan en pantalla un *frame* de cada X reales generados por el juego. Muchos emuladores actuales de no ser por esta opción funcionarían a cámara lenta y serían inviables para casi todos los usuarios que poseen máquinas de prestaciones escasas.

Como último recurso, en caso de que no se pueda conseguir más velocidad de ningún otro sitio, se puede optar por recortar características como las siguientes: eliminar scrolls parallax en caso de que haya muchos, bajar características de sonido o música, bajar el número de *sprites* que puede haber en pantalla simultáneamente, etc.

El proyecto MAME

Como hemos dicho, MAME es uno de los proyectos más ambiciosos del sector de la emulación de recreativas.

MAME son las siglas del inglés *Multiple Arcade Machine Emulator* y el proyecto fue iniciado por Nicola Salmoira hace muchos meses.

MAME, como ya se ha dicho anteriormente, no es un proyecto cerrado de un programador y nos encontramos que muchas personas están aportando ampliaciones para que el mismo soporte nuevos ROMS (nuevos juegos).

En la actualidad MAME soporta más de 120 juegos diferentes, incluyendo prácticamente todos los clásicos de las primeras épocas (PCMAN y similares) y está dirigido actualmente por Mirko Buffoni. La popularidad que está adquiriendo este emulador está haciendo que aparezcan versiones del mismo en otras plataformas, como son las ya existentes Macintosh, Commodore Amiga (aunque su rendimiento en cuanto a velocidad de ejecución se refiere es muy bajo todavía) y varias versiones UNIX. En la actualidad podemos decir que este emulador está en pleno auge y prácticamente cada semana aparece una versión nueva del mismo que incorpora soporte para nuevos juegos.

MAME es el proyecto más ambicioso dentro de la emulación de recreativas

Para aquellos que estén interesados en este emulador desde el punto de vista de programación, les informamos de que los fuentes originales están disponibles para cualquier interesado y que pueden ser bajados en la WEB oficial de MAME, que es la siguiente **<http://lim.dsi.unimi.it/curr/mix/mame/index.htm>**.

Casi todos los emuladores actuales requieren como mínimo un Pentium para funcionar correctamente

Como último detalle a señalar, destacar que una versión del MAME, la de UNIX, la ha escrito un programador español, lo cual le da más interés al proyecto en general.

Para más información sobre MAME existen infinidad de páginas dedicadas a él que tocan el tema, ya sea exclusiva o parcialmente, y en el listado 2 damos los principales *links* disponibles.

Cómo aprender más

Como es lógico, aunque tengamos mucha información de toda clase de hardware, conozcamos las técnicas básicas y demás, para el programador clásico es casi indispensable que pueda conseguir fuentes (que los hay) de emuladores ya creados y documentos tipo "Tutorial" y "FAQ" sobre programación que podamos encontrar y que harán las veces de libro de aprendizaje.

En la WWW hay infinidad de textos técnicos sobre muchas plataformas

Aunque para encontrar este tipo de documentos tendremos que pasarnos más horas delante de nuestros buscadores preferidos y canales IRC, normalmente vale la pena el esfuerzo. Así, por ejemplo, si queremos adentrarnos en la emulación de máquinas Mac basadas en arquitectura 680x0, nada mejor que bajarse el documento que habla sobre el funcionamiento interno del EXECUTOR, un conocido emulador MAC, y con tan sólo leerse y entender bien el documento, que está localizado en la dirección **<http://www.ardi.com/MacHack/machack.html>**, tendremos muchos conceptos, datos e ideas, incluso con ejemplos, que nos podrán servir de mucha ayuda a la hora de comenzar .

Listado 3. Ejemplo de código Mac 680x0 y su equivalente en 80x86.

Código original del 680x0:

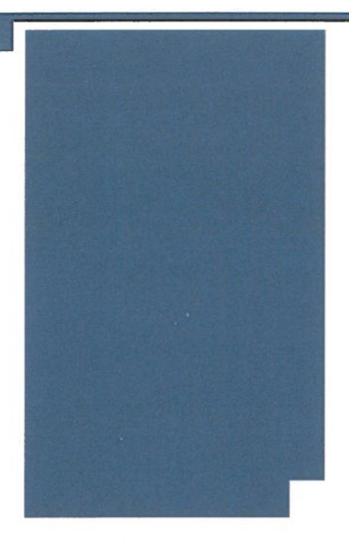
```
movel #0,d2
moveb d0,d2
lslw #8,d0
orw d0,d2
movel d2,d0
swap d2
orl d2,d0
movel a0,d2
lsrb #1,d2
bcc 0x3fffed4
```

Código equivalente del 80x86:

```
xorl %ebx,%ebx  ; movel #0,d2
movl _d0,%edx   ; moveb d0,d2
movb %dl,%bl
shlw $0x8,%dx   ; lslw #8,d0
orw %dx,%bx     ; orw d0,d2
movl %ebx,%edx ; movel d2,d0
rorl $0x10,%ebx  ; swap d2
orl %ebx,%edx   ; orl d2,d0
movl _a0,%ecx   ; movel a0,d2
movl %ecx,%ebx
shrb %bl ; lsrb #1,d2
movl %ebx,_d2   ; <almacena 68k
movl %edx,_d0   ; registros a memoria>
jae 0x3b734c    ; bcc 0x3fffed4
jmp 0x43d48c    ; <regreso al código 68k>
```

Conclusión

Con lo explicado en estas páginas se dispone de todo lo necesario para que el no iniciado en el tema pueda empezar a organizar un proyecto propio de emulación. Como es lógico, no se han explicado técnicas de programación para implementar ningún emulador concreto para ninguna máquina específica, puesto que ello ocuparía varios libros y no procede. Antes de acabar, sólo dar las gracias a Moisés Ferrán por su colaboración en la búsqueda de URLs.



Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

Pregunta

Tenemos un problema con una dirección en la WEB. La primera vez que accedimos se estaba cargando una imagen y como tardaba mucho, paramos con el STOP la transferencia. A partir de entonces, cada vez que intentamos acceder la imagen no se carga completamente, se para justo donde cortamos la trasferencia y aparece el mensaje "DOCUMENT DONE". Los *proxies* que tenemos puestos en la configuración del Netscape y que no podemos cambiar ¿pueden ser la causa del problema?, ¿existe algún tipo de herramienta que asegure que se obtenga realmente la información solicitada?

Respuesta

El problema que planteas debería solucionarse pulsando sobre el botón 'Recargar' o *'Reload'*. Esto debería funcionarte ya que en casi todos los navegadores los mandatos del tipo Recargar o Renovar (depende del navegador) son incondicionales, es decir vuelven a cargar la página independientemente de que ésta se halle ya en la caché de archivos o no. Si no te funciona asegúrate de tener activada la opción para cargar también imágenes en tu navegador.

Si todo esto no funciona parece que tienes un problema con la caché de archivos de tu navegador. Lo más seguro es que la fecha de tu ordenador sea incorrecta, de modo que los archivos del directorio de caché aparecen siempre como los más modernos y se cargan en todas las ocasiones. La caché de archivos va guardando los documentos e imágenes de los sitios visitados en un directorio local (de tu propio ordenador) de modo que la próxima vez que se visite la carga del documento sea mucho más rápida, ya que la realiza desde el propio disco duro y no desde el servidor. De esta manera se ahorra tiempo y (sobre todo) dinero en la factura de Telefónica. Esta técnica es muy útil ya que casi todos acabamos visitando siempre las mismas páginas .

Lo que hace el navegador es comparar las fechas de los archivos que están en la caché y cargar la versión que tiene en el disco duro local, si la fecha es igual, o cargar la versión que tiene el servidor, si es más moderna que nuestra versión local. Como parece que tu problema viene de este lado y no nos aportas muchos datos sobre la configuración de tu navegador, elegiremos esta: borrar todos los archivos de la caché, de ese modo el navegador no tendrá donde elegir a la hora de cargar los archivos. En el caso de Netscape para configurar el caché de archivos hay que elegir *Preferencias de la red de red* del menú *Opciones*, elegir la pestaña *Caché* y pinchar el botón *Borrar el caché de disco ahora*. De esta manera se borran los archivos que el navegador va guardando para agilizar la carga de documentos. También se puede borrar directamente el contenido del directorio destinado a caché de disco mediante comandos del sistema operativo o cualquier utilidad de gestión de disco. Para estar totalmente seguro de que el documento cargado es efectivamente el que está en el servidor, se desactiva la caché de disco. Con ello la carga de documentos es más lenta, pues el navegador no cuenta con los archivos del disco local para agilizar la carga del documento. Sin llegar a tanto, hay opciones de caché configurables. Son:

- Caché del disco: se especifica el tamaño del caché del disco en kilobytes. La opción *Borrar el caché del disco ahora* vacía el caché inmediatamente.
- Directorio del caché de disco: se especifica el directorio donde se guardarán los archivos para caché del disco.
- Verificar documentos: se especifica con qué frecuencia se comprueba la red para buscar versiones actualizadas de los documentos. Al comprobar si hay documentos actualizados, se recupera (si es necesario) la página actualizada del servidor de la red y no una página potencialmente antigua de la caché.
- *Una vez por sesión:* el navegador comprueba si hay páginas actualizadas solamente una vez durante una sesión.
- *Cada vez:* se comprobará si hay cambios cada vez que pida ver una página, con lo que la carga de documentos será lenta.
- *Nunca:* no se realizará ningún tipo de verificación.

CONTENIDO DEL CD

Nueva versión del Java Development Kit, Java WorkShop 2.0, Proxys para Windows 95 y NT.

En este número estrenamos interfaz, la instalación es sencilla, sólo hay que pulsar los botones de la aplicación y seguir las instrucciones del *wizard* instalador.

Java

Presentamos la última versión desarrollada en los laboratorios SUN, no hay muchos cambios con respecto a las anteriores pero conviene estar actualizado. Además del *Java Development Kit* (Kit de desarrollo) hemos incluido las versiones actualizadas del *Runtime Environment* (entorno de ejecución) y el *Performance Pack*,lo que facilitará el trabajo a los desarrolladores más exigentes. Como Microsoft no quiere perder el tren de Java también actualiza sus versiones, la ultima versión de su *Software Development Kit* está también incluido en el CD, con ella y el Visual J++ no habrá applet java que se nos resista.

Hay también otros desarrolladores de software verdaderamente dignos de mención como es el Java Workshop en su versión 2.0, como siempre la última.

Proxys

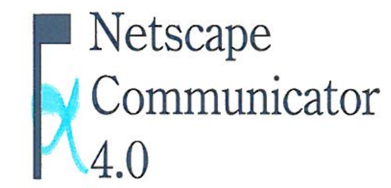
Este mes hemos prestado especial atención a la oferta de Proxys que hay en el mercado, parece claro que la proliferación de intranets, extranets e internet ha producido este crecimiento y necesitamos de un software serio que proporcione al administrador de redes la seguridad en el intercambio de información entre las redes y además acelere su funcionamiento, éstas son las principales funciones actuales de un Proxy Server, la de FireWall o Corta fuegos entre dominios y la de Cache de direcciones y paquetes entre los distintos nodos y con la red exterior.

Recomendamos que echéis un vistazo a las versiones de un Proxy que hacen los distintos fabricantes.

Demonios para NT

Poco a poco te vamos proporcionando el suficiente software para crear tu propio proveedor de servicios de internet, o cuando menos para que tú en tu casa te familiarices con la instalación y uso de los diferentes servicios o demonios que utiliza Windows NT para controlar el correo de los usuarios de ese servidor o los grupos de noticias que alberga.

Instálalos en tu servidor, verás que es fácil y que no necesitas una preparación especial.



Netscape Communicator 4.0

Este fabricante de software no se conforma con un simple browser de web, y ofrece una verdadera suit con toda clase de aplicaciones para realizar las tareas que puedas imaginar en internet. En cada nueva versión los dos fabricantes siempre intentan sacar algo que el competidor no haya sacado todavía, lo que hace que la actualización de estos programas sea continua y dolorosa para nuestro bolsillo pero mejora sin ninguna duda la calidad de estos programas. No dudes en instalártelo.



Visual Basic

La revolución (qué tópico) ha llegado a la hora de confeccionar aplicaciones cliente servidor, si quieres ver cómo lo hacen los profesionales échale un vistazo a la Demo de Crescent que hemos incluido y a los diferentes componentes Visual Basic y ActiveX que la acompañan; presta atención a esto último, ActiveX, ya es casi omnipresente en Internet, parece que microsoft está consiguiendo lo que quería y todo el mundo está migrando sus aplicaciones para soportar esta arquitectura.